EXAMENSARBETE INOM DATATEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, SVERIGE 2016

Återställningsverktyg för fordon baserat på applikationsintegration

Vehicle Restoration Application based on Enterprise Application Integration

AMITA BHUDDI

OLIVER SOMOS

KTH
SKOLAN FÖR TEKNIK OCH HÄLSA

VRAP

Vehicle Restoration Application based on Enterprise Application Integration

Återställningsverktyg för fordon baserat på applikationsintegration

Författare

AMITA BHUDDI

OLIVER SOMOS

Examensarbete inom Datateknik
Grundnivå, 15 hp
Handledare på KTH: Anders Lindström
Examinator: Ibrahim Orhan
TRITA-STH 2016:61

KTH Kungliga Tekniska Högskolan
Skolan för Teknik och Hälsa
136 40 Handen, Sverige

Sammanfattning

Mjukvarutestningsgruppen på Scania använder en manuell och tidskrävande process för att återställa testfordon efter sina utförda arbeten. I den återställningsprocessen ingår ett antal olika applikationer för att kunna säkerställa att fordonet är i samma skick som det var innan testerna. För att förbättra arbetsflödet med minskad arbetsbelastning och att försäkra en säkrare process var de intresserade av att utveckla ett återställningsverktyg som kan göra detta uppdrag.

Önskemålet var att skapa detta genom att återanvända så mycket av de redan tillgängliga komponenterna som möjligt. Då det fanns flera komponenter som uppfyllde funktionskraven för de flesta funktionaliteten krävdes det en undersökning för att avgöra vilka ska användas. Denna gjordes med hjälp av integrationsmodellen Enterprise Application Integration, där målet är att utveckla en slutprodukt av applikationerna som används inom en organisation för att förenkla bland annat underhåll, datahantering och utbildning av medarbetarna. En prototyp har tagits fram som implementerar tre existerande moduler på olika nivåer och som enligt målet med EAI själv är en enkel mjukvara som möjliggör att komponenterna tillsammans utför arbetet.

Nyckelord

EAI Enterprise Application Integration, integrationsmodeller, modularitet, återställning

Abstract

The software testing team at Scania use a manual and time consuming process to restore a test vehicle after working with it. Several different applications are used in this process to ensure the vehicle is in the same state as it was before their testing. To improve the workflow with a reduced workload and a more robust process, the test team was interested in the development of a restoration application.

It was desired to develop the restoration application by reusing the components to the greatest possible extend. Since there were many components that fulfilled the needs of most functions, a pre-study of all the applications was done to decide which components can be re-used. This was study was based on the integration model, Enterprise Application Integration, which aims to create a single product combining the applications used in an organization to simplify processes such as maintenance, data management and employee training. A prototype was developed which implements three existing modules on different levels and, in line with the goals of EAI, is itself a simple application that enables the components to work in unision.

Keywords

EAI Enterprise Application Integration, integrations models, modularity, restoration

Förord

Denna rapport har skrivits som en del av kursen *Examensarbete i programmet Datateknik* på Kungliga Tekniska Högskolan under andra hälften av vårterminen 2016. Arbetsuppgiften var framtagen av mjukvarutestningsgruppen YSPV på företaget Scania AB för att förenkla deras arbete under testningsprocessen.

Vi vill tacka handledaren Anders Lindström på skolan för arbetet med att sätta sig in i uppgiften så bra. Vi är tacksamma för all stöd och värdefull feedback som vi fick under examensarbetets gång gällande skriftliga rapporten och själva utförandet av arbetet.

Vi vill även passa på att tacka vår handledare Rovan Luca på Scania, som har hjälpte oss att komma i gång med Scanias verksamhet. Rovan har alltid varit insatt i återställningsprodukten och kom med nya idéer och förbättringsförslag för uppgiften. Engagemanget av gruppchefen Maria Nygren och de andra í testningsgruppen har också hjälp oss att förbättra slutprodukten vid olika tillfällen. Ansvariga för applikationerna från andra avdelningar som blev intervjuade vill vi också tacka för värdefull feedback under förstudien.

Innehållsförteckning

Ordlista

API, Application Programming Interface
Kan översättas till applikationsprogrammeringsgränssnitt. Detta gränssnitt är en del av integrationsmodellen enligt EAI.

CAN, Controller Area Network
CAN används bland annat för att styra funktioner som körriktningsvisare, bromsljus och överföra data från tändningen, reglage, etc. till fordonets centrala dator.

Demofilen
Filen som sparar tillstånd av fordon för att komma åt fordonets data utan att koppla sig mot fordon i off-line läge.

DTC, Diagnostic Trouble Codes
Felkoder som finns lagrade i styrenheter, som beskriver fel som kommer från systemen ECU:n övervakar. Kan läsas ut så att felet kan åtgärdas, och tas bort.

EAI, Enterprise Application Integration
En kombination av processer, mjukvaror, hårdvaror eller olika standarder som resulterar till ett eller flera integrerade system som kan operera som en enda produkt.

ECU, Electronic Control Unit
En styrenhet för styrning av olika system och funktioner i fordon, som kan bestå av en hårdvara och en programvara.

FPC, Functional Product Characteristic
FPC block finns i SOPS-filen som beskriver fordonets fysiska konfiguration. Blocket innehåller FPC koder med dess utförande som är nödvändiga för att parametersätta fordonets elsystem.

SDP3, Scania Diagnos & Programmer 3
Diagnos- och programmeringsverktyg för felsökning och mjukvaruuppdatering av ett fordons elsystem, som ingår i Scanias specialverktygssortiment.

SOPS, Scania Onboard Product Specification
En fil som finns i fordon, industrimotorer, samt databaser i Scania, som beskriver produktens elektriska system. Den innehåller uppgifter som gör det möjligt att programmera elsystemet.

Testrigg
En uppsättning av styrenheter sammankopplade på samma sätt se de skulle vara i ett fordon. De finns i kontoret så att testerna kan utföras utan att man har tillgång till en riktig lastbil.

VCI, Vehicle Communication Interface
Ett gränssnitt för att kommunicera med ett fordon som fungerar som en mellan-hand mellan en dator och fordon.

VIN, Vehicle Identification Number
Varje fordon har ett unikt identifieringsnummer så att det är enklare att identifiera.

XML, eXtensible Markup Language
Ett märkspråk för att definiera datastrukturer i dokument, så att det kan skrivas och läsas på ett väldefinierat sätt.

1 Inledning

Det här kapitlet börjar med en kort beskrivning om Scania som företag och avdelningen där examensarbete har utförts. En kort beskrivning om problemet anges vidare med eventuella målsättningar för examensarbete. Avslutningsvis beskrivs vissa avgränsningar som ingår i arbetet.

1.1 Bakgrunden

Scania är ett globalt företag och är känd som en ledande tillverkare av tunga lastbilar, bussar och industri- och marinmotorer. Dessutom tillhandahåller och säljer företaget ett stort utbud av tjänst relaterade produkter och finansiella tjänster. Scanias verksamhet är spridd över ett hundratal länder med mer än 44 000 anställda [1].

Forskning och utveckling på företaget är koncentrerad till Sverige. Tillverkning sker mest i Europa och Latinamerika med möjlighet till globalt utbyte av såväl komponenter som kompletta fordon. Företagets kärnvärde är “kunden först”, “respekt för individen” och “kvalitet”. Detta reflekteras bra med Scanias identitet bland dess kunder, anställda och deras värderingar och arbetsmetoder.

YSPV, som är beteckningen för gruppen Quality, Integration and Test, ansvarar för att säkerställa kvalité av olika applikationer som används i eftermarknaden i olika område gällande styrenheter och beståndsdelar som krävs för produktombyggnad. Integration och verifiering görs mest för produkter som används av verkstäder och andra som arbetar med fordonet. Verifiering i detta sammanhang är att se till att produkten är väl designerad enligt kraven för att leverera all funktionalitet till kunden. De vanliga kunderna som YSPV jobbar med i dagsläget är främst verkstäder som jobbar med Scania Diagnos & Programmer 3 (SDP3) och Conversions, men också komponentutvecklare inom interna verksamheten samt metod ingenjörer och systemägare på Forskning och Utveckling avdelningen.

1.2 Problemformulering

I de flesta fall är det önskvärt att fordonet eller testriggen befinner sig i samma tillstånd efter utfört arbete, bland annat testning, som före. Med tillstånd menas fordons läge med alla inställningar och data. Samma sak gäller för testriggar också som är en uppsättning av styrenheter som liknar fordon så att det går att utföra tester enklare utan fordon. Därför krävs det att de som under sitt arbete ändrar på konfiguration av styrenheter återställer den tidiga konfigurationen. Idag görs detta med en manuell och tidskrävande process, där flera olika program måste användas för att återställa fordontillstånd. I detta examensarbete kallas den här funktionaliteten återställning så att ett fordons eller en riggs tillstånd kan sparas och återställas vid testningsprocess.

Förutom att det kostar tid, som skulle kunna användas för annat medans verktyget utför återställningen, är det inte heller självklart att alla som ändrar på styrenheternas konfiguration som en del av sina arbetsuppgifter kan använda alla de program som ingår i processen. Resultatet av detta är att fordon och testriggar för ofta lämnas i ett icke önskvärt skick som försvårar arbetet för nästa person som skulle behöva använda dem.

Eftersom den här processen använder sig av specifika funktioner från flera applikationer för att hantera återställning, ingår utvärderingar och integrationen av de befintliga applikationerna med en lämplig lösning för en slutprodukt i examensarbetet. Själva slutprodukten är tänkt att användas endast inom Scania vilket gör att det bara är användbart inom företaget. Men själva principen, det vill säga, applikationsintegration är något som kan vara intressant för större publiken.

1.3 Målsättning

Examensarbetets övergripande målsättning är att utveckla ett återställningsverktyg för Scania. Applikationsintegration för slutprodukten kommer att baseras på applikationer som används internt inom Scania för att göra manuella återställningsprocessen till automatiserad, som kan användas i en liknande situation.

Följande delmål ska uppnås:

- Målgruppens behov ska analyseras utifrån önskade funktioner, arbetsflöde och integration för slutprodukten.
- Att ta reda på vilka interna applikationer som kan vara lämpliga för att tillämpa applikationsintegration i verksamheten ska undersökas.
- Att förstå ingående de valda applikationernas fungerande och integrationsmöjligheter för slutprodukten. Med integrationsmöjligheter menas de olika nivåer som integration kan baseras på.

- Återställningsverktyget ska vara lätt anpassbart för nya produkter eller situationer inom företag genom att basera olika delar modulärt.
- Utifrån undersökningen av applikationerna, ska ett lösningsförslag presenteras och tillämpas som innehåller
  - Flödet för data sparandet och återställningsprocessen
  - Hur installation går till, där det är speciellt viktigt hur delkomponenterna hämtas
  - Hur uppdateringar av delkomponenter hanteras
  - Filstruktur för sparad data och loggar
- En integrationsprototyp ska tas fram som kan utföra hela data sparandet och återställningsprocessen. Prototypen ska utvärderas i verklig arbetsmiljö.

1.4 Avgränsningar

Projektarbetet har begränsade resurser vad gäller tid, omfattning och kunskap.

- Tidsmässigt är projektet begränsat till tio arbetsveckor, därför är det viktigt att göra en prioriteringslista enligt krav från företaget tidigt. Detta tankesätt gör det enklare för företaget när det gäller skötsel av produkten framöver.
- Examensarbetet kommer inte att täcka enhetstestning för att det är en tidskrävande process och helt nytt för författarna. Det var ingenting som uppdragsgivaren kräver eller prioriterar.
- Kalibreringsvärde för styrenheter i fordon kommer inte att tas upp i återställningsflödet. Detta för att i dagsläget är det oklart vilka styrenheter som innehåller kalibreringsvärde och vilka verktyg kan läsa de värden ifrån och skriva tillbaka till dem till styrenheterna. Slutprodukten kommer inte att inneha den funktionaliteten på grund av tidsbrist.

Eventuell överbliven tid i det fall att dessa mål hinns med kommer att ägnas till att förbättra produkten med avseende på ytterligare funktioner som kan ingå i sidomål som projektgruppen tar fram löpande under arbetet.

2 Teori och bakgrund

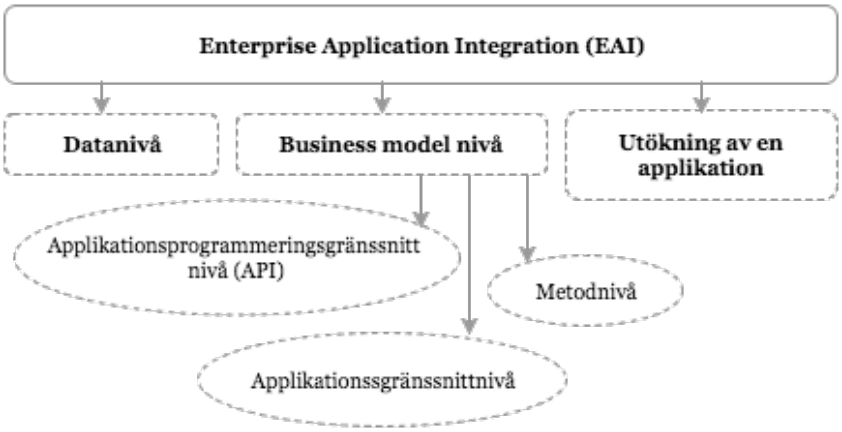
Kapitlet börjar med en beskrivning om vad affärsapplikationsintegration, förkortas EAI är. Vidare tas upp applikationsintegration på olika nivåer som sammanfattar hur integrationen kan uppdelas. Fördelar och nackdelar med integrationstekniker beskrivs också vidare i första avsnittet av det här kapitlet. Det finns några utmaningar under integrationsprocessen som avhandlas vidare. Nästa avsnitt går ut på att beskriva tidigare arbeten som har utförts inom samma område gällande integrationen.

2.1 Enterprise Application Integration

Enterprise Application Integration eller affärsapplikationsintegration, förkortas till EAI, är en koncept som kom fram under 90-talet. Kärnkonceptet i EAI är att integrationen kan möjliggöras med så lite programmering och kostnad som möjligt med hjälp av de redan befintliga applikationerna i verksamheten. En kort och koncis beskrivning av EAI ges av J. Lee, m.fl. [2] enligt följande.

> *"EAI is a business computing term for plans, methods, and tools aimed at modernizing, consolidating, and coordinating the overall computer functionality in an enterprise."*

EAI kan delas i olika typer beroende på vilken nivå och hur applikationer integreras. Figuren 2.1 visar de olika typerna som beskrivs mer ingående i nästa avsnitt.



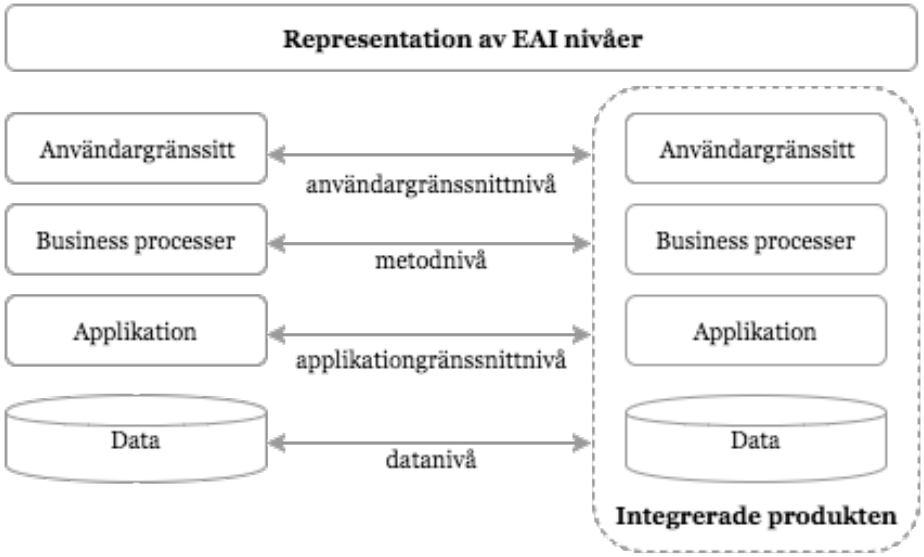
Figur. 2.1 De olika nivåerna av applikationsintegrationsmodellen

I en verksamhet brukar det finnas olika applikationer eller databaser som används för att fullborda ett arbete. EAI idén bygger på att fortsätta använda dem, lägga till eller migrera vissa funktioner enligt behov till nya applikationer. När det gäller förbättringar kan man tolka det på olika sätt; det kan vara nya teknologier, nytt synsätt på applikationshantering, nya sätt att öka produkteffektivitet, etc.

Nuförtiden ses EAI också som ett gemensamt gränssnitt lager för olika applikationer att kommunicera med varandra. Den här applikationsintegrationsmodellen ger en möjlighet att utveckla en implementering på en business-objekt nivå på flera sätt [2]. För det första finns det en möjlighet för utökning av dataintegration inom ett gemensamt ramverk. Det ger stöd för länkning mellan data och processer på applikationsnivå. Dessutom kan fördelning av affärslogik på komponentnivå i verksamhet uppnås. Slutligen implementeras förbättringsmöjligheter för användargränssnitt som anses som grunden för integrationen i flesta situationer. EAI kan delas i olika typer beroende på vilken nivå och hur applikationer integreras.

2.1.1 Typer av integration

Det finns olika typer av EAI beroende på hur integrationen tillämpas. Typerna eller nivåerna beskriver hur nära de olika delarna knyts ihop i den resulterande applikationen. Figuren 2.2 beskriver representation av EAI nivåer med tillhörande lager.



Figur 2.2 En representation av EAI nivåer

Nedan finns några typer av integration.

Datanivå

Applikationer använder delad data i form av gemensamma filer eller databaser, där de olika komponenter kan spara information som andra kan arbeta vidare med efteråt. Här ingår också eventuell kommunikation mellan de databaserna eller gemensamma lagringsplatser [3]. Integration på en data-nivå kräver extra uppmärksamhet då chansen att duplicera data och funktionaliteter brukar vara högre än de andra nivåerna [8].

Business modell nivå

Den här nivån kan också kallas för kod-nivå integration som är baserad på komponent-per-komponent idén. Här integreras de bästa delarna av befintliga applikationer till slutprodukten [8]. Den kan variera på olika nivåer baserad på applikationsnivå, metodnivå eller användargränssnittnivå och beskrivs nedan [3].

- **Applikationsprogrammeringsgränssnittnivå (API)**

  Här används applikationsprogrammeringsgränssnitt för att stödja integrationen för den slutliga lösningen. För att använda API:er krävs det att applikationen har ett sådant gränssnitt och att gränssnittet är tillgängligt. Det innebär också att en del av logiken måste finnas i den nya koden som använder sig av gränssnittet. Utvecklarna brukar använda applikationsgränssnitt för att få tillgång till information eller olika processer för att integrera applikationer. API-nivån föredras mest nuförtiden [4].

- **Användargränssnittnivå**

  Delarna är lösast kopplade på den här nivån. Det gemensamma är endast gränssnittet mot användaren. Komponenterna används i sina helheter. Den här typen av integrationen gör också att integritet av olika delarna behålls oavsett vilket sammanhang de används eller återanvänds i efterhand. Tillgång och underhållning av de delarna går mycket smidigare än i de andra nivåerna.

- **Metodnivå**

  Metoder från de färdiga applikationerna återanvänds utan någon typ av ändring alls. Den här nivån delar på business-logik bland olika applikationer med hjälp av ramverksanvändning inom ett företag.

Utökning av en befintlig applikation

Att välja en av delapplikationerna och utöka den med resten av funktionerna som krävs är också ett möjligt tillvägagångsätt. Fördelen är att organisationen får en komplett applikation som inte har några beroenden, som kan vara fördelaktigt på grund av bl.a. enklare underhåll. Utökning har dock nackdelar också, som ett större arbete under integrationsprocessen och ett byte från befintliga system som ersätts av den nya.

2.1.2 Fördelar med EAI

Det finns vissa fördelar med användning av integration av applikationer. Några av fördelar listas nedan.

- **Effektivitet**

  Om många separata system används inom företaget är risken stor att de kommer att delvis uppfylla samma funktion, som leder lätt till att data som hör ihop sparas på olika platser, i olika format, eller flera gånger [8]. Det slösar både lagringsutrymme och försvårar uppsökning av information. Sker all datainmatning genom samma system blir det mycket enklare att undvika dessa problem.

- **Resurser**

  Genom att återanvända komponenter tar det mindre tid och pengar att utveckla lösningen än om det hade byggts om från grunden, men ger samtidigt samma fördelar [8].

- **Förenklad produkt**

  På grund av att det bara är en slutprodukt som kommer att användas förenklas många processer inom organisationen, bland annat utbildning för medarbetarna, hantering av rättigheter och tillgång, installation och uppdatering på medarbetarnas datorer.

2.1.3 Utmaningar

När man applicerar EAI konceptet i en verksamhet finns det flera utmaningar som ingår och bör ta hänsyn till innan produktutveckling kan göras med hjälp av integrationsmodellen. Summering av de utmaningarna beskrivs nedan.

- **Komplexitet**

  EAI tekniken i sig bygger på att integrera olika delar som i sin tur ökar komplexitet av slutliga produkten för att både utvecklarna och förvaltarna bör förstå olika delar av systemapplikationer. Mer än bara förståelse ingår det även en genomtänkt lösning för att integrera dem. En bra EAI-tillämpad produkt har bra uppdelning av delar och modularitet så att det är enkelt att byta ut vissa delar eller ändra dem på ett relativt simpelt sätt.

- **Långsiktigt åtagande**

  Beroende på vilka delar som används från befintliga applikationer kan EAI betyda ett långsiktigt åtagande för vissa verksamheter. Frågan om hur andra applikationer fungerar, vilka stöd de olika delapplikationerna kräver, hur uppdateringar ser ut, etc. är några av omständigheterna [8]. En bra EAI-modell ska räkna med alla sådana delar vid utveckling och planering av slutliga produkten.

- **Prestanda**

  EAI kommunikationen, i vissa sammanhang, kräver en hög prestanda beroende på vilka applikationer som används som stöd. Denna faktor bör tas hänsyn till vid utveckling och återanvändning [5].

- **Utveckling både vertikalt och horisontellt**

  Här gäller det att tänka på skalbarhet av den integrerade applikationen som baserades på EAI principen. För att tänka på hela verksamheten kan man prata om utveckling på båda axlar, vertikal och horisontell [5]. När det gäller horisontell axel, avser man generalisering av slutprodukten och när det gäller vertikala axeln tänker man mer på andra begränsade resurser som kunskaper och tid. Med detta är utmaningen att hitta så bra lösningen som möjligt som kan anpassa med de ovanstående axlarna.

- **Applikation- och nätverkssäkerhet**

  I de flesta fallen är säkerheten på olika nivåer en stor fråga. Det för att undvika de säkerhetsluckorna som kan uppstå vid integrationen eller kan bli farligare under integrationsprocessen. Säkerheten beror på vilken nivå integrationen sker och vilka applikationer som integreras med anknytning till krav från företag. Om integrerade applikationen används internt inom ett företag behöver man inte tänka särskilt mycket på säkerhet och tvärtom om det är för extern användning.

2.2 Tidigare arbeten

Själva applikationen är tänkt att återanvända de befintliga delarna som används inom företaget i dagsläget. Detta för att möjliggöra vidareutveckling samt enklare underhåll. Tonvikten i detta examensarbete ligger på att hitta den bästa möjliga lösningen för att integrera de olika segmenten från de existerande verktygen hos Scania. Det finns ett antal tidigare arbeten inom området affärsapplikationsintegration. Principen för den här delen är att sätta in arbetet i ett större sammanhang och för att undersöka vad som redan är gjort inom området.

2.2.1 Utmaningar och framtida möjligheter av applikationsintegration

Soomor och Anwar [6] lyfter fram idén med EAI, Enterprise Application Integration, det vill säga en kombination av processer, mjukvaror, hårdvaror eller olika standarder som resulterar till ett eller flera integrerade system som kan operera som en enda produkt. Detta med anknytning till ökning till effektiviteten i verksamhet, att standardisera olika tillvägagångssätt som finns till applikationer och att forma nuvarande behov och återanvända tekniken till framtida behov.

Olika typer av EAI tas upp vidare i publikationen bland annat data-, applikation-, metod- eller användargränssnitt lager. Soomor och Anwar tar också upp problem och utmaningar som även vi kan inträffa under integrationsprocessen. Det tänkbara problemet som man stötar på i början är att om integrationsstrategi inte är tillräckligt genomtänkt kan det påverka utveckling på ett negativt sätt. Det för att själva arbetet bygger på att använda befintliga produkter till ett större behov på så effektivt sätt som möjligt. Det andra problemet när det gäller EAI-baserade applikationer är att negligera aspekter som säkerhet, prestanda eller övervakning. Lösningsförslag och tillämpning av de olika användbara teorimodellerna bidrar stort till detta examensarbete när det gäller EAI implementation. Mer detaljer i kapitel 3.

2.2.2 Fördelar med affärsintegrerade system

Arbetet, Benefits of Enterprise Integration Systems [7], ger en tydlig bild av hur stor betydelse rätt implementering av EAI kan betyda för en organisation. En diskussion om EI, Enterprise Integration, antyder vissa förmåner som en verksamhet kan ta nytta av genom att integrationen bidrar stort till produktivitet. Detta fungerar som en sammankoppling mellan olika system, moduler eller applikationer som bidrar till en bättre relation mellan lager struktur i ett system. Det gör att EAI är mer produktiv än de traditionella sätten att utveckla applikationer. När det gäller återställningsarbete gäller det att hantera EI, Enterprise Integration, på ett rätt sätt så att istället för att skriva om koden kan man välja att integrera vissa användbara delar av olika applikationer. På så sätt kan man spara tid och resurser och lägga de resurserna istället på ett smartare integrationsval och implementering.

2.2.3 Lösningar inom integrationsområdet

I arbetet, Existing Approaches to Software Integration [8] nämner Land och Crnkovic tre olika approacher till mjukvaruintegration såsom data-, kodnivå eller genom att utöka ett system. Mer förklaring av de lösningarna tas upp i nästa avsnitt 2.2.1. Land diskuterar i sitt doktorandarbete, An Architectural Approach to Software Evolution and Integration [9] diskuterar djupare kring dessa lösningar och resonerar kring alla tre på en djupare nivå. När det gäller återställningsverktyg kommer integration implementeras på data- och kodnivå lösningar beroende på lämpliga biståndsdelar av utvecklingsprocessen som beskrivs i Kapitel 3. Här är det intressant att jämföra alla lösningar på olika nivåer samt hur effektivt dem är i verkligheten och hur de kan anpassas för utveckling av integrerade produkten som detta examensarbete handlar om.

2.2.4 Modularisering och EAI

Arbetet, Värdeskapande genom modularisering [10] är baserad kring modularisering och att tänka modulärt. Scania som företag lägger en stor tonvikt på att tänka och utveckla modulärt [11]. Modulariseringen anses mer som en företagsstrategi som skapar värde i tillverkande företag. Detta möjliggörs på grund av tydliga gränssnitt i produktarkitektur.

Det gör att i efterhand är det smidigt att vidareutveckla produkter på grund av minskad produktkomplexitet och möjliggöra återanvändning av moduler för eventuella framtida behov. Under utvecklingsprocessen ska modularisering prioriterats så gott det går. MFD, Modular Function Deployment, konceptet har förklarats väl av Gabriella Björk i sitt arbete i olika verksamheter med fokus på produktion. Men detta koncept är lika användbart i mjukvarusammanhanget. Sammanfattningsvis gör det enklare att underhålla produkten efteråt och enklare att baseras återanvändning av applikationsmoduler.

2.3 Återställningsverktyg

De fyra funktionerna som återställningsverktygsprototyp är tänkt att innehålla enligt uppdragsgivarna beskrivs nedan under avsnitt 2.3.3 till 2.3.6. Vilka komponenter som räknas som ett tillstånd för ett fordon eller en rigg förklaras först i avsnitt 2.3.1 för att underlätta förståelsen för de huvudfunktionerna. Även förklaring av termer som flashprogrammering och reservdelsprogrammering som nämns under beskrivning för huvudfunktionerna tas upp under 2.3.2.

2.3.1 Tillstånd

När man pratar om tillstånd av ett fordon eller en testrigg ingår vissa komponenter. En kort förklaring av vad de komponenterna som SOPS-filen, styrenheternas data och demofilen innebär beskrivs nedan.

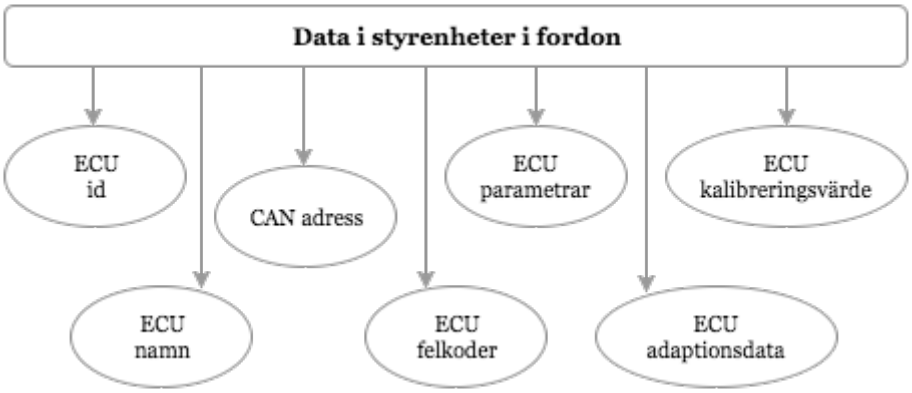
SOPS-filen

En förkortning för Scania Onboard Product Specification. SOPS-filen kan jämföras med fordonets DNA som beskriver fordonets konfiguration sett ur ett elsystemsperspektiv. Den innehåller uppgifter som gör det möjligt att programmera elsystemet genom att parametersätta fordonets styrenheter. SOPS-filen sparas vanligtvis i två förutbestämda styrenheter i fordonet. Filen läses i första hand från den ena styrenheten. Ifall filen saknas där kan den läsas ifrån den andra styrenheten som säkerhetskopia enligt Scanias interna dokumentation av SOPS-filen.

Det finns ganska mycket information om fordon i SOPS-filen men när det gäller själva återställningsverktyg används bara två delar från SOPS-filen. Den ena är en så kallad FPC, Functional Product Characteristic, 1-värde, som beskriver typ av fordon. Exempelvis, om värdet är “A” är fordonet en lastbil, “B” om det är en buss och “F” om det är en marinmotor. Detta värde används för att skriva tillbaka SOPS:en till rätt enhet under återställningsfas. Den andra delen från SOPS-filen är VIN, det vill säga Vehicle Identification Number eller identifieringsnummer, som är unikt för varje fordon och är ett sätt, ur utvecklingsperspektiv, att veta från vilket fordon tillstånd sparas ifrån.

ECU data

Varje ECU, Electronic Control Unit eller styrenhet, har viss information som finns i fordonet vid produktion. Figuren 2.3 ger en översikt av all data som hamnar under den här kategorin.



Figur 2.3 Bilden visar vad termen ECU-data innebär

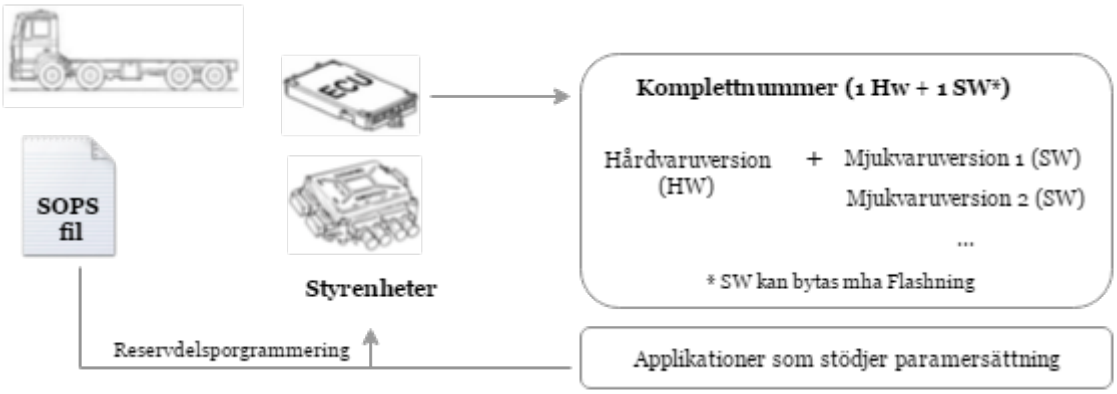
Det här datat finns bland annat i form av ECU ID eller identifieringsnummer, CAN-adress, ECU namn, ECU DTC eller felkoder, adaptionsdata, kalibreringsvärde och parametrar. ECU identifieringsnummer eller komplettnummer, som kan förkortas till ID, är ett identitetsnummer för varje styrenhet beroende på vilken hårdvaru- och mjukvaruversion den har. Varje ECU har ett namn och CAN-adress för att kännetecknas bland alla styrenheter som finns i ett fordon. För varje enhet finns det också en s.k. felkodsbeskrivning i form av DTC, Diagnostic Trouble Codes som används eventuellt för att diagnostisera de felen i styenheterna. Adaptionsdata är en unik typ av data för vissa ECU:ar som samlas under körning. Kalibreringsvärden är också unik för vissa styrenheter, och innebär värden som måste ställas in för varje fordon efter att det är färdigbyggt. Ett exempel på kalibreringsvärde är data i styrenheten som är kopplad till en lutningssensor.

Med parametrar syftas olika inställningar på styrenheterna som finns i fordon. Antal inställningar och vad de innebär är olika för varje styrenhet. En ECU-parameter exponeras också mot omgivningen genom skriv- och lästjänster. ECU-parametrar kan sättas på två olika sätt. Det första sättet är göra en parametersättning manuellt med hjälp av interna verktyg som används inom verksamheten. DevelopmentTool, som beskrivs nedan, är ett av dem, där värden bestäms fritt av användaren. Det andra sättet är så kallad reservdelsprogrammering, där sätts alla parametrar i alla styrenheter enligt en Scania databas och SOPS-filen. Båda sätt kan hanteras via verktyget.

Demofilen

Filen som sparar tillstånd av ett fordon eller en rigg för att komma åt fordonets eller riggens data utan att koppla sig mot fordon i off-line läge.

Sammanhanget



Figur 2.4 Bilden visar sammanhanget av olika komponenter i ett fordon

Bilden ger en översikt på var de komponenterna befinner sig i på ett riktigt fordon. Ett fordon innehåller många beståndsdelar i form av styrenheter. Varje styrenhet som finns i ett fordon har sitt eget komplettnummer som är en kombination av en hårdvaruversion och en mjukvaruversion. Flashning kan användas för att byta de mjukvaruversionerna som passar till hårdvaran. Utbyte av mjukvaruversion utan att byta hårdvara är också ett koncept som kan anknytas till modularitet. SOPS-filen som är sparad i förutbestämda styrenheterna används för reservdelsprogrammering av andra styrenheterna för att parametersätta. I övrigt kan parametersättning också göras via andra applikationer som de har stöd för styrenheterna. Mer om flashprogrammering och reservdelsprogrammering tas upp i nästa avsnitt.

2.3.2 Andra termer

Flashprogrammering

Flashning eller flashprogrammering betyder inladdning och installation av en styrenhets mjukvara på hårdvaran. Det är helt enkelt ett sätt att versionsstyra mjukvaran i olika styrenheter som finns i ett fordon. Vanligtvis har varje hårdvara flera mjukvaror, och en databas används för att hålla reda på möjliga mjuk- och hårdvarukombinationer. Efter flashningen tappar styrenheter all sparad information, som parametrar, eventuell SOPS-fil, etc. och därför måste parametersättningen och återskrivningen av SOPS-filen utföras efter flashning i återställningsflödet. Mer beskrivning av hur flashningen användas för att fullborda återställningsfunktion finns i kapitel 3.

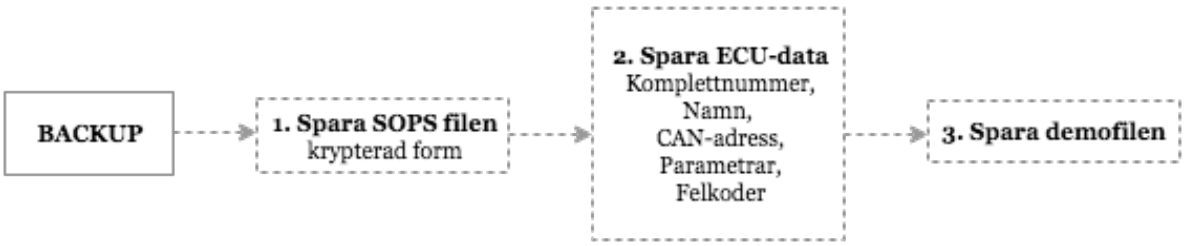
Reservdelsprogrammering

Reservdelsprogrammering innebär att alla parametrar av en styrenhet sätts till ett förbestämt värde. Värdet kommer från en databasfil som installeras tillsammans med några av verktygen som beskrivs i nästa avsnitt eller finns tillgängligt på Scanias servrar. En SOPS-fil som passar fordonet används för att avgöra vilka egenskaper styrenheten ska ha och värdet för de egenskaperna hämtas från databasen.

Reservdelsprogrammeringsprocessen kan också påverka SOPS-filen som därför ska skrivas tillbaka till förutbestämda enheterna i fordonet efteråt. Databasens innehåll styrs av en ansvarig grupp på Scania och kan inte ändras manuellt. Databasens versionsuppdateringar är dock viktiga för att innehållet kan uppdateras regelbundet med värden som anses vara bättre anpassade eller efter några buggfixar.

2.3.3 Backup

De grundläggande stegen som borde ingå i backup flödet visas i figuren nedan. SOPS-filen och viss data från styrenheterna behöver sparas av processen för att kunna återställa fordonet till detta tillstånd vid ett senare tillfälle. Utöver det ska en demofil också sparas som kan användas inom arbetsgruppens andra aktiviteter.



Figur 2.5 Bilden visar det tänkbara backupflödet

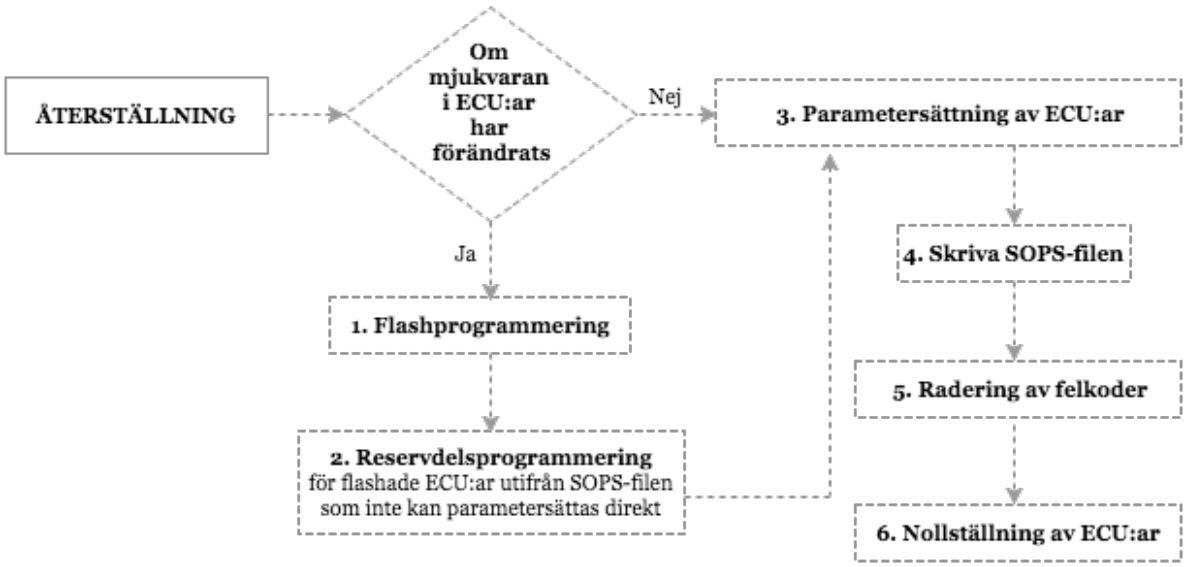
2.3.4 Reservdelsprogrammering

Reservdelsprogrammering är en enklare process som det visas i Figur 2.6. SOPS-filen läses från fordonet, sedan används för att parametersätta styrenheterna enligt en databas. När parametersättningen är klar skrivs tillbaka SOPS-filen till fordonet då det är möjligt att processen påverkar den.

Figur 2.6 Bilden visar det tänkbara reservdelsprogrammeringsflödet

2.3.5 Återställning

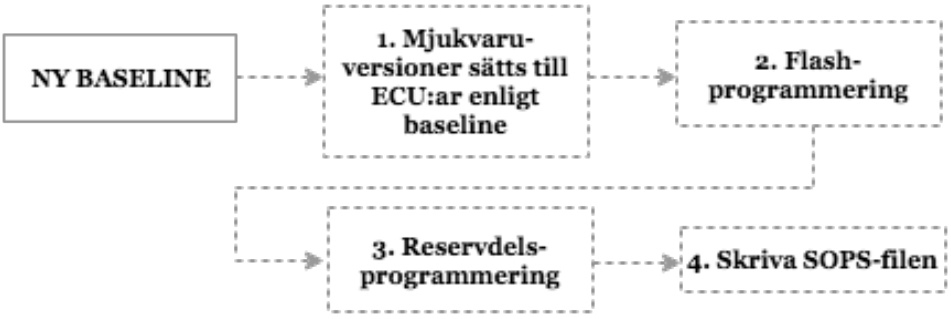
Figur 2.7 visar den önskade återställningsprocessen. Den utgår från ett tidigare skapad tillstånd. Här ska det först avgöras om mjukvaran i styrenheterna stämmer eller om de behöver bytas. Efter det ska parametrar skrivas tillbaka till enheterna, och sist ska SOPS-filen skrivas tillbaka till fordonet.



Figur 2.7 Bilden visar det tänkbara återställningsflödet

2.3.6 Ny baseline

Mjukvaran i styrenheterna sätts till önskad version genom flashprogrammering och sedan parametersättas styrenheterna på samma sätt som i reservdelsprogrammering som beskrevs i kapitel 2.3.4, med skillnaden att SOPS-filen är inte hämtad från fordonet utan vald av användaren.



Figur 2.8 Bilden visar det tänkbara backup-flödet

2.4 Undersökta verktyg för integration

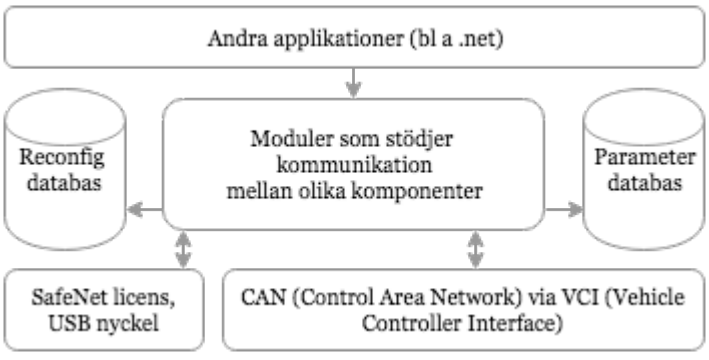
I detta avsnitt beskrivs de olika verktygen som undersöktes under förstudien för att bestämma vilka som kan ingå i integrationen för återställningsverktyget. Eftersom alla verktygen används internt inom Scania hölls namnen och vissa beskrivningar av dessa verktyg hemligt. Resonemang kring de delarna som används för produktutveckling och integration anges. Komponenterna består av programapplikationer, databaser eller eventuella bibliotek som kan kopplas eller återanvändas till återställningen av fordon. Tabellerna i bilaga 3 jämför också de olika funktionerna.

2.4.1 FlashDB

FlashDB är en databas som håller reda på möjliga kombinationer av styrenhetens hård- och mjukvara, som det nämndes i flashningsdel i avsnitt, 2.3.2. Eftersom en ECU:s hårdvara stödjer bara vissa mjukvaruversioner är det oerhört viktigt att mjukvaruversionen stöds av hårdvara för att uppfylla flashningskrav. Det finns övrig information i form av flaggor som exempelvis visar att på vilka sätt och hur kan flashning stödjas för de ECU:arna. Några exempel på flaggor är stöd för reservdelsprogrammering, test, produktion, etc.

2.4.2 MainLibrary

MainLibrary är ett bibliotek som har vissa användbara metoder för att stödja återställningsfunktion för att hantera styrenheter i fordon. Vissa diagnosapplikationer som används i verksamheten för att hantera styrenheterna tar nytta av detta bibliotek för att utföra sina funktioner. Biblioteket gör också möjligt att använda diagnoskommunikation för några eftermarknadsprodukter som Scania Diagnos & Programmer 3 (SDP3), produktionsverktyg, etc.



Figur 2.5. Bilden visar hur MainLibrary används

Figur 2.5 ger en översiktlig förklaring på hur denna tjänst fungerar. Den gör en koppling mellan parameter-databas och en reconfig-databas internt samt ser till att licens via USB nyckel finns och hanteras som det ska.

Det ansvarar också för kommunikationen mellan CAN, Controller Area Network och datorn via en s.k. VCI, Vehicle Communication Interface. På grund av modularitet kan andra applikationer byggas enkelt på det här biblioteket. Det för att det är en samling av funktioner som kan anropas av de överliggande applikationerna.

2.4.3 Flasher

Detta bibliotek används av flera olika verktyg för att flashprogrammera styrenheter i fordon. I flashningsprocessen ingår kontroll mot databasen, FlashDB för att avgöra om önskad flashning är möjlig samt att kontrollera om användaren har de nödvändiga rättigheterna för att utföra flashningen. Förutom de andra verktygen som implementerar biblioteket finns det även ett väldigt enkelt program som är bara ett grafiskt gränssnitt för dessa funktioner för att manuellt kunna flasha styrenheterna. Det är detta program som används i den manuella återställningsprocessen idag.

2.4.4 FlashLibrary

Biblioteket används för att utföra flashprogrammering men på en lägre nivå än Flasher. Det används av Flasher, MainLibraryGUI och flera andra interna utvecklingsapplikationer i grunden för att utföra flashprogrammering av styrenheter på ett fordon på Scania.

2.4.5 SOPSHandler

SOPSHandler är ett simpelt verktyg som med endast ett kommandoradsgränssnitt kan läsa, spara och skriva SOPS-filen från och till ett fordon eller en rigg. Det är också möjligt att göra en reservdelsprogrammering via detta verktyg. Verktyget var framtaget av en utvecklargrupp för att förenkla flödet av interna automatiserade tester av produkten som de utvecklar. Detta verktyg använder sig också av MainLibrary för att möjliggöra båda funktionerna.

2.4.6 DevelopmentTool

Det här är ett internt verktyg inom Forskning och Utveckling avdelningar som bland annat används för att ändra på parametrar och titta på felkoder. Det stödjer också diagnos- och EOL-programmering. Det kan däremot inte spara eller skriva SOPS-filen. När det gäller parametersättning, är verktyget väldigt kraftfullt för att det inte har begräsningar på värden som kan sättas. Användaren får se förslag på värden, men den får själv skriva in vad som helst.

DevelopmentTool är dock endast avsedd för styrenheter som är utvecklade av Scania, som innebär att många av styrenheterna kan inte hanteras. Verktyget har också ett kommandoradsgränssnitt som stödjer en väldigt liten del av funktionalitet, nämligen att spara alla parametrar från styrenheter till olika filer, alltså en fil per enhet och att skriva filen tillbaka, som råkar vara just det som behövs för återställningsverktyget.

2.4.7 MainLibraryGUI

Det är också ett internt verktyg med lite andra möjligheter än vad DevelopmentTool erbjuder. Det kan läsa och skriva SOPS-filen, all ECU data inklusive ID, parametrar, etc. går att läsa men man kan inte ändra på parametrar godtyckligt. Det kan parametersätta endast genom reservdelsprogrammering. Läsning och borttagande av felkoder är också möjligt med verktyget som är intressant ur återställningssynvinkel. DevelopmentTool fungerar bra när det gäller parametersättning av ECU:ar men med en nackdel att det inte stödjer alla ECU:ar som finns i ett fordon. Det stödjer bara Scania tillverkade ECU:ar som nämns ovan också men MainLibraryGUI kan parametersätta alla enheter.

2.4.8 VersionChecker

Ett gränssnitt mot FlashDB för att manuellt kunna avgöra om en hård- och mjukvarukombination på en styrenhet är giltig eller inte. Det här verktyget används ibland i den manuella processen för återställning av fordontillstånd. Kombinationer visas upp med hjälp av en graf som visar mjukvarustöd för en hårdvara i en form av en kedja. Det finns andra funktionaliteter som visar kategori, eller om den ECU:n stödjer reservdelsprogrammering och andra saker som kan vara viktiga ur produktionssynvinkel. När det gäller återställningen så begränsas användning av detta verktyg bara för att kontrollera mjuk- och hårdvarukombination för flashningen. Eftersom återställning görs från de sparade gamla mjukvaruversioner, behöver VersionChecker inte användas. Denna applikation användes främst vid manuell återställning.

2.4.9 ECUtool

Det här verktyget används för att fullfölja samma funktionalitet som Flasher, det vill säga göra flashprogrammering. Det kan också användas för att läsa och skriva SOPS-filen från styrenheter i fordon. Dessutom kan det också sätta ECU parametrar via reservdelsprogrammeringen med hjälp av SOPS-filen.

3 Metod

I detta kapitel kommer en redogörelse för vilka metoder som användes under examensarbete. Användarundersökning i form av intervjuer och undersökningsmetodiken genom att kartlägga användning av olika komponenter är några förförande som ingår under metodik. Testning och utveckling av prototypen ingick också som metodik för att utvärdera den valda lösningen. Därefter beskrivs vilka applikationer som kan återanvändas för att fullborda återställningsarbete och hur integrationen av olika applikationer kan göras på bästa möjliga sätt. Olika funktionaliteter som verktyg bör innehålla, enligt kravlistan förklaras och detaljer om utvecklingen skildras vidare.

3.1 Användarundersökning

Denna del gjordes i början av arbetet för att få en större förståelse för vilka användare som skall använda produkten i första hand. Genom att ha en bredare bild av användarna kan man se vad den gruppen verkligen behöver. Detta gav möjligheter till projektarbetarna att knyta nya kontakter samt att förstå kraven mer ingående. Eftersom återställning av fordontillstånd inkluderar olika delar används i dagsläget flera applikationer för att fullborda denna funktionalitet manuellt. Detta på grund av att olika verktyg kan göra vissa delar och en lösning för det är Enterprise Application Integration, EAI. För att säkerställa att alla applikationsintegration och återanvändning av moduler kan ske så effektivt som möjligt började arbetet med att kontakta olika utvecklingsgrupper.

3.2 Undersökningsmetodik

Förstudien började med intervjuer av representanter från olika team som har utvecklat de applikationer som kommer att integreras i återställningsverktyget. Intervjuerna i stort sett var avsedda för att få en helhetsbild och möjliga kopplingar vid automatisering av återställningsprocessen. Frågor som ställdes till alla utvecklarna som intervjuades finns i Bilaga 1. De delarna som fokuserades mest under intervjuerna var följande.

- **Uppbyggnad**

  Här ingick frågor om tekniska detaljer i de olika applikationerna som kan integreras. Uppbyggnadsfrågor täckte också funktioner som är specifika för just de applikationerna och om det går att möjligen använda dem via gränssnitt. Diskussion om möjliga förslag från olika utvecklingsgrupper med avseende på återställningsfunktion togs också upp.

- **Integrationsmöjligheter**

  Det var viktigt att få veta vilka gränssnitt de befintliga applikationerna har som gör integration möjligt och på vilket sätt. Beroende på vilka funktioner som finns i olika applikationer, intervjuades utvecklingsgrupper om eventuella problem som kan uppstå vid integrationsprocessen.

- **Modularitet**

  Det var viktigt att utveckla slutprodukten modulärt så att vid behov kan vissa moduler bytas ut utan stort bekymmer. Med modularitet i det här sammanhanget menas att olika funktioner som efterfrågas i kravställning från kunden har modulärt tankesätt så att ändring av delarna kan göras enkelt utan att påverka andra funktioner i återställningsprocessen.

- **Användningsfall**

  För att förstå funktioner som slutprodukten ska ha djupare och effektivare användes metodiken av att dela upp de olika funktionerna i användningsfall. På så sätt kan man förstå flödet av alla delfunktioner på ett enklare sätt som dessutom gör utvecklingsprocessen enklare att följa. Användningsfallen för de olika funktionerna finns tillgängliga under Bilaga 2 av rapporten.

- **Övrigt**

  Det visade sig också att andra avdelningar är intresserade av återställningsverktyget. Detta måste tas hänsyn till i vissa delar av applikationen, bland annat att olika grupper kommer behöva ha olika platser för sparande av fordonsdata, så för att enkelt kunna ändra på det måste det finnas inställningsmöjligheter. På grund av att det är många som kommer förmodligen att använda verktyget borde ett enkelt gränssnitt med för användaren tydliga funktioner finnas så att det krävs minimal träning för användaren.

3.3 Tekniska detaljer

Följande avsnitt kommer att beskriva de olika applikationerna som möjligen kunde integreras för varje funktion i återställningsverktyget. För att jämföra olika applikationer med varandra kan man också referera till tabellerna i bilaga 3.

3.3.1 Val för SOPS-hantering

För att läsa och skriva SOPS-filen från och till fordon fanns det olika förslag under förstudien efter intervjuer från utvecklingsgrupper av de befintliga applikationerna. De förslagen nämns följande.

- **SOPSHandler**

  SOPSHandler via ett kommandoradsgränssnitt var ett alternativ för att läsa och skriva SOPS-filen från och till fordonet. Den här typen av integrationen hamnar under användargränssnittnivå för att återanvändning av applikationen skulle ha varit direkt via ett anrop. En nackdel med användning av SOPSHandler är att då läggs det till ytterligare ett verktyg som slutprodukten blir beroende av. Det skulle tydligen vara ett problem som kan komplicera saker när det skulle gälla uppdateringar av slutprodukten. Återställningsverktyget var tänkt från början att inte behöva uppdateringar så länge de underliggande verktygen inte ändrar på sina gränssnitt. Eftersom de underliggande applikationer uppdateras kontinuerligt internt inom företaget, är det generellt bättre att minska antalet komponenter som ingår i återställningsprocessen.

- **MainLibrary**

  Användning av MainLibrary skulle vara en integration på applikationsprogrammeringsgränssnittnivå (API-nivå) som var andra alternativet. Fördelen med användning av MainLibrary är att det förenklar beroenden för slutprodukten till färre applikationer eller komponenter då detta bibliotek måste användas för andra funktioner också. Det i helheten ger bättre kontroll över återställningsflödet. Nackdelen med att använda MainLibrary är att det kräver större arbetsinsats vid utvecklingen av slutprodukten så att återanvändning av biblioteket görs rätt.

- **ECUtool**

  Det visades under förstudien att ECUtool kan användas för SOPS-filens läsning från respektive styrenheter. Det kan också användas för att skriva tillbaka SOPS-filen till fordon. Det skulle möjligen också vara en integration på användargränssnittnivå. Här gäller också det återkommande problemet med uppdateringar av ECUtool om det valts att använda för återställningsprocessen.

3.3.2 Val för ECU-data hantering

Några val för att hantera ECU data finns nedan.

- **MainLibrary**

  Användning av MainLibrary skulle också betyda att hantering av ECU-data bland annat identifieringsnummer, CAN-adresser för styrenheter och namnläsning skulle bli enklare än att använda sig av något annat verktyg för slutprodukten. Det skulle också betyda en smidigare kommunikation direkt med fordon och dess styrenheter.

- **DevelopmentTool**

  När det gäller läsning av ECU data som krävs för återställningsverktyg, kan även DevelopmentTool användas. Data i detta fall inkluderar bara av namn och komplettnummer men inte CAN-adress från styrenheterna som finns i fordon eller testriggar. Användning av det verktyget för integration i slutprodukten skulle betyda användargränssnittnivå i EAI modellen. Den här funktionen är inte direkt tillgänglig via kommandoradsgränssnitt. För att komma åt ECU-namn och komplettnummer behöver man parsa dem från XML filer som sparas av DevelopmentTool.

3.3.3 Val för ECU-felkodshantering

För att radera och spara felkoder fanns följande två förslag under förstudien.

- **MainLibrary**

  Detta bibliotek förutom kommunikation med styrenheter kan eventuellt användas för att radera och spara felkoder från enheter och även göra en direkt nollställning. Den enda svårighet med användning av biblioteket var att användaren får ingen beskrivning av DTC:er som är en förkortning för Diagnostic Trouble Code eller felkoder i styrenheter, som är viktigt om användaren vill läsa av felkoden och vill veta mer om hur de felkoderna kan diagnostiseras.

- **DevelopmentTool**

  I skillnaden från MainLibrary finns det en bättre hanteringsmöjlighet i DevelopmentTool när det gäller felkoder i fordonet styrenheter. Det för att mer information finns om felkoder kan skrivas till en fil via DevelopmentTool som kan vara en stor nytta för användaren om det behövs under diagnostikprocessen för att felsöka och rätta de felkoderna i fordonet. DevelopmentTool kan bara läsning och radering av felkoder från styrenheter i fordonet.

3.3.4 Val för ECU-parameter hantering

För att hantera ECU-parametrar fanns det bara ett val, nämligen följande.

- **DevelopmentTool**
  För att hantera parameterläsning vid backup och skrivningen vid återställning kan DevelopmentTool vara ett bra val. DevelopmentTool kan installeras helt separat från återställningsverktyget och dess kommandoradsgränssnitt används för att få tillgång till funktionaliteten. Detta kommer att vara en integration på användargränssnittnivå.

  Efter diskussion med bland andra handledaren på Scania har arbetsgruppen kommit fram till att det blir bäst om parametrarna i de styrenheter som DevelopmentTool inte stödjer ignoreras. På grund av att det enda andra sättet att skriva parametrar är reservdelsprogrammering anser uppdragsgivaren att återställningsprocessen blir bättre om de icke stödda enheterna bara ignoreras. Om och när stöd till dem läggs till i DevelopmentTool kommer den automatiskt finnas i återställningsverktyget på grund av typen av operationer som används. När parametrar sparas anropas DevelopmentTool med ett kommando som sparar parametrar i en fil per enhet för alla stödda enheter som finns i fordonet, och vid återskrivning går programmet igenom alla sparade filer och skriver de tillbaka till fordonet. På det sättet spelar det ingen roll för återställningsverktyget hur många av styrenheterna stöds.

3.3.5 Val för flashprogrammering

För varje styrenhets hårdvara finns det olika versioner av mjukvara som en användare kan skriva till. För att göra den här skrivningen av en mjukvaruversion till styrenheterna används flashprogrammeringskoncept inom företaget. De följande tre applikationer används i dagsläget för flashprogrammering finns nedan.

- **Flasher**
  Detta bibliotek var ett av alternativen för att uppföra flashning för styrenheter. Användning av biblioteket skulle innebära integration på applikationsgränssnittnivå enligt EAI modellen. Bibliotek används inom företaget som ett användargränssnitt för att underlätta testning och kan hantera flashning för bara en styrenhet åt gången. Efter intervjun med utvecklingsgruppen av detta bibliotek fick man veta att implementeringen inte skulle vara så svårt. Samtidigt får man möjlighet att få hjälp med utveckling av slutprodukten genom att få tillgång till användning av vissa funktioner från biblioteket. Ifall Flasher skulle användas för återställningsverktyg kommer uppdateringar av biblioteket inte påverka slutprodukten. En till fördel när det gäller användning av det här diagnosbiblioteket är att det underlättar integrationen ganska mycket.

- **ECUtool**

  Det här verktyget som nämnts i kapitel 2 används också för att fullfölja samma funktionalitet som Flasher, det vill säga göra flashningen. Implementering av skulle möjligen också vara en integration på användargränsitt nivå. Här gäller också det återkommande problemet med uppdateringar av ECUtool om det skulle användas för återställningsprocessen.

- **FlashLibrary**

  FlashLibrary är också en tänkbar lösning för flashprogrammering av styrenheter som ovanstående två applikationer. Skillnaden med användning av biblioteket är att det gör flashning på en lägre nivå för enheter som ger mer kontroll men samtidigt förminskar enkelhet för implementeringen i återställningsverktyget.

- **DevelopmentTool**

  DevelopmentTool innehåller också en komponent som kan flashprogrammera styrenheter och den är även tillgänglig via kommandoradsgränssnittet, men den komponenten är inte anpassad för att användas på fordon. Detta innebär att den inte kan användas i återställningsverktyget för att utföra flashprogrammering eftersom återställning kommer att göras både på riggar och på fordon.

3.3.6 Val för reservdelsprogrammering

För att reservdelsprogrammera styrenheterna kan följande applikationer användas.

- **MainLibrary**
  MainLibrary kan uppfylla många av funktionerna som krävs av återställningsverktyget och reservdelsprogrammering för styrenheter är en av dem. På grund av det innebär det att en API-nivå integration ger användandet av detta bibliotek bättre kontroll över processen än ett färdigt verktyg på bekostnad av utvecklingstid. Som nämnts ovan skulle användning av MainLibrary vara en bra lösning för många andra funktioner som förenklar kommunikationen med fordon.

- **SOPSHandler**
  Som det har diskuterats ovan var SOPSHandler ett alternativ för SOPS hantering som kan utföra reservdelsprogrammering också. Nackdelen med att utnyttja SOPSHandler är att det blir ett beroende för slutprodukten som måste tas hänsyn till vid framtida uppdateringar. Fördelen är att genom att använda det blir integration enklare och kortare. Implementeringen kommer återigen vara på en användargränssnitt nivå då SOPSHandler används via ett kommandoradsgränssnitt.

- **ECUtool**
  Reservdelsprogrammering var en funktionalitet som också erbjöds av ECUtool via ett kommandoradsgränssnitt. Implementationen på användargränssnitt nivå är enklare än det första alternativet, MainLibrary och detta verktyg ger en bättre kontroll över processen än ECUtool. Men en nackdel som visade sig under förstudien var den svåra hanteringen av användarrättigheter.

3.3.7 Övrigt

Övrigt att ta upp angående de befintliga applikationerna.

- **MainLibraryGUI**

  Det fanns en applikation till som används på företag för att hantera SOPS-filen och läsa parametrar från styrenheter. Den kan också göra reservdelsprogrammering av styrenheterna utifrån SOPS-filen. Förutom det kan applikationen även läsa andra data från styrenheterna bland annat namn, identifieringsnummer och felkoder. MainLibraryGUI är egentligen ett grafiskt gränssnitt för MainLibrary. Verktyget har inget kommandoradsgränssnitt för att återanvända de befintliga funktionerna. Därför kan slutprodukten inte integreras på användargränssnittnivå.

3.4 Utvecklingsmetodik

Följande avsnitt beskriver vissa detaljer om utvecklingen av slutprototypen och vilken metodik som användes vid utveckling. En redogörelse av hur integrationen ser ut för slutprodukten görs först som följer med beskrivning av applikationer som användes under integrationen och att slutföra de fyra funktionerna till slut.

3.4.1 Modularisering

Modulariseringen, som uppges tidigare i rapporten, är en företagsstrategi som skapar värde i tillverkande företag. Den ger smidighet att vidareutveckla produkter på grund av minskad produktkomplexitet och möjliggöra återanvändning av moduler för eventuella framtida behov. Examensarbetets huvuduppgifter förutom att välja rätt implementering utifrån integrationsmodellen enligt EAI, var också att göra integrationen på ett modulärt sätt.

Slutprodukten har tillämpning av fyra funktioner, nämligen sparande av fordon data, återställning, göra reservdelsprogrammering och att sätta upp en ny baseline för fordon. Vid implementering är återställningsverktyget uppbyggt så att funktionerna kan lätt utbytas vid behov i framtiden. Att funktioner är utbytbara innebär också att nya applikationer kan användas istället för just de applikationerna som har valts för utveckling av slutprodukten. På så sätt blir testning av de funktionerna också enklare än om de inte var modulariserade.

3.4.2 Utvecklingsmiljö

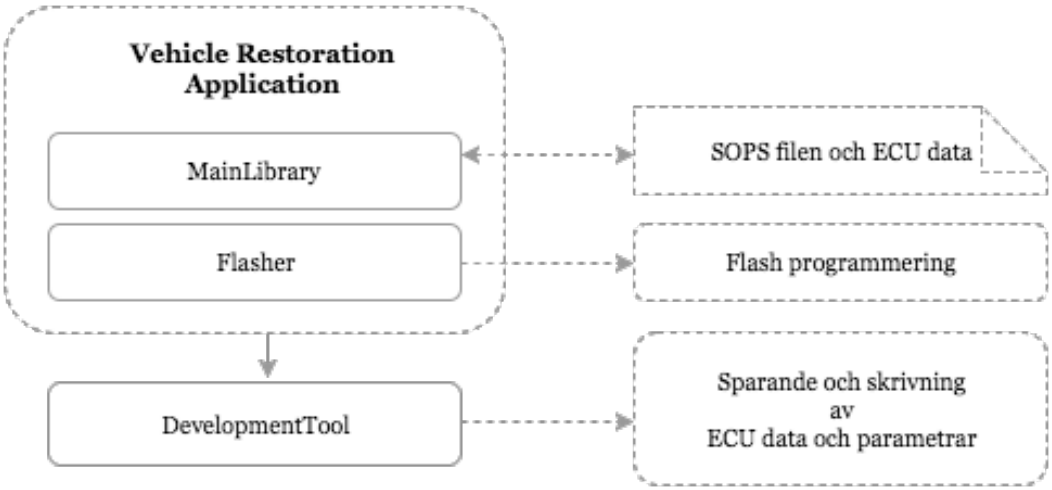
På grund av att båda bibliotek som används under utveckling av slutprodukten är skrivna i programmeringsspråket C#. Därför valde produktutvecklarna samma programmeringsmiljö för att utveckla återställningsverktyget. Microsoft Visual Studio Professional 2012 används som produktutvecklingsmiljö för produktutveckling. Dessutom blir underhållning av återställningsverktyg i efterhand enklare på grund av C# som språket. Dessutom hade produktutvecklarna möjlighet att få mycket hjälp från utvecklarna på företaget.

4 Resultat

I det här kapitlet börjar med att beskriva hur systemprototypen är uppbyggd och hur integrationen kopplas till EAI-modellen. En sammanfattning av vilka komponenter används för slutprodukten uppges vidare.

4.1 EAI och systemprototyp

I kapitel 2.1 togs upp vad Enterprise Application Integration, EAI, är och på vilka nivåer kan det tillämpas. När det gäller återställningsverktyg kan integrationsprocessen kort sammanfattas med Figur 4.1. MainLibrary och Flasher biblioteken används direkt i prototypen och DevelopmentTool används via ett anrop från prototypen. Vilka nivåer de komponenterna tillämpas på förklaras nedan.



Figur 4.1 Återställningsverktygets komponenter

- **Applikationsprogrammeringsgränssnitt nivå (API-nivå)**

  För att läsa SOPS-filen och läsa ECU data i form av ECU ID, också kallas för komplettnummer och felkoder från olika styrenheter används MainLibrary. Den här integrationen är på applikationsprogrammeringsgränssnittnivå också kallad API nivå. Som beskrivs ovan används SOPS-läsning och skrivning i alla fyra funktioner, nämligen Backup, Återställning, Reservdelsprogrammering och Ny baseline. ECU data används under Backup, Återställning och Ny baseline. Användning av MainLibrary är gjort på en API-nivå för att slutprodukten använder metoder från detta bibliotek för att kommunicera med fordon, tillsammans med SOPS-fil hanteringen med ECU data.

När det gäller flashning används Flasher vilket hamnar också under API nivå på business modell integrationsnivå. Anledning för detta val är också att metoder från detta bibliotek används direkt i slutprodukten enligt API-nivån. Figur 4.1 beskriver användning av biblioteket via slutprodukten som gör att användaren kan göra flashprogrammering.

- **Användargränssnitt nivå**

  DevelopmentTool används på användargränssnitt nivå för att spara och skriva tillbaka de sparande parametrarna till ECU:ar. Figur 4.1 visar att DevelopmentTool ligger utanför slutproduktens uppbyggnad för att slutprodukten gör ett anrop till ett gränssnitt för DevelopmentTool. I detta fall används ett kommandoradsgränssnitt för att återanvända funktionaliteter som finns i DevelopmentTool. Därför blir den här integrationen på en användargränssnittnivå i business modell uppdelning av EAI.

4.2 Valda komponenter

För hantering av SOPS-filen ansågs MainLibrary vara det bästa alternativet. För hantering av annat ECU data som enheternas namn, CAN-adress och unikt komplettnummer har det inte funnits några alternativ heller. Som SOPS-filens läsning och skrivning passade dock MainLibrary väldigt bra här. I prototypen används MainLibrary för att hantera felkoder från styrenheterna. Det fanns inga alternativlösningar när det gällde sparandet och skrivningen av styrenheternas parametrar förutom DevelopmentTool. Därför valdes det som det självklara valet för att hantera parametrarna.

För programmering av mjukvaran i styrenheterna vid en befintlig hårdvaruversion i fordon, det vill säga flashprogrammering, valdes Flasher som bästa alternativet. För reservdelsprogrammering valdes också MainLibrary då det används redan för andra funktioner. Anledning till val för olika komponenter bakom funktioner tas upp under kapitel 5, Analys och diskussion.

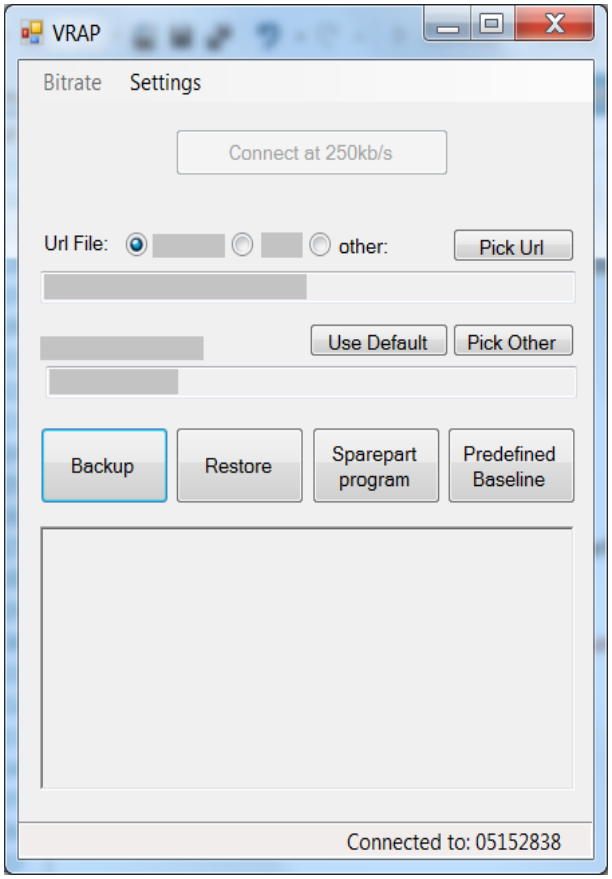
4.3 Systemprototyp

I detta avsnitt beskrivs systemprototypen för återställningsverktyg. Första delen ger en översikt av hur alla funktioner implementeras till en applikation. Beskrivning av alla funktioner som prototypen innehåller kommer i nästa del.

4.3.1 Grafiska gränssnittet

Återställningsverktygets prototyp är en Windows Forms applikation som knyter ihop de tre applikationerna som har valts för att integreras i slutprodukten.

Figuren 4.2 visar utseendet av slutprodukten med alla fyra funktioner som ingår i återställningsflödet. De större knapparna, *Backup*, *Restore*, *Sparepart program* och *Predefined baseline* utför de huvudfunktionerna sparandet, återställning, reservdelsprogrammering och uppsättning av en ny baseline för fordon. Djupare förklaring av de funktionerna finns i nästa avsnitt.



Figur 4.2 Återställningsverktyg

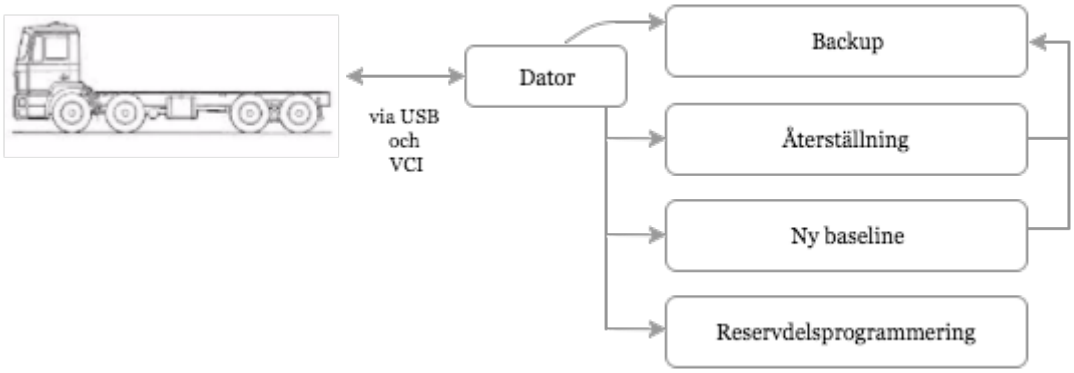
Under menyn *Bitrate*, finns det inställningar på vilken frekvens verktyget ska kopplas till fordonet. I vanliga fall räcker det med 250 kb/s bithastighet men med pågående utveckling av nya fordon på företaget kan den bithastigheten öka till 500 kb/s eller mer. Slutprodukten ger den möjlighet också för användarna.

Under menyn Settings, finns det inställningar som default sökväg där dialogrutan för att välja mapp och filer i de olika funktionerna ska öppnas. På så sätt behöver användaren inte söka om sökvägen vilket kan bli jobbigt på grund av den komplexa filstrukturen. Även sökvägen för DevelopmentTool som används för sparandet av parametrar finns med i denna meny.

Vissa namn och delar i figuren är grå markerade enligt uppdragsgivarens önskemål eftersom de används internt på företaget för att stödja fungerandet av flashprogrammering och reservdelsprogrammering. Den stora textboxen efter de fyra funktionerna är uppsatt för att visa aktiviteter med tiden för användaren under olika processer. Idén är att samma text skrivs till loggfilen så att användaren kan spåra de eventuella felen som kan uppstå under körningen. I nedre hörnet visas fordonets identifieringsnummer som återställningsverktyget är uppkopplad mot så att det är enklare att identifiera fordon vid mapp val.

4.3.2 Funktionaliteter

Här beskrivs de tänkta processflöden för återställningsverktygets funktioner i prototypen. De är i stor del framtagna av Scania och utgår från den manuella processen, med vissa detaljer och förslag från författarna. Först görs en utgångspunkt med hjälp av "Backup" funktion som sparar fordonets tillstånd. Efteråt kan man med "Återställning" funktionen återskapa ett godtyckligt sparat tillstånd till fordonet. Reservdelsprogrammering är också en funktion som kan ingå i återställningsprocessen, men den blev en separat funktion efter feedback från uppdragsgivaren. "Ny baseline" är en funktion som har tillkommit under utvecklingsarbetet. Den bygger på återställningsfunktionen men kräver inte en fullständig utgångspunkt.



Figur 4.3. Bilden visar funktioner i återställningsverktyget

Figuren 4.3 visar att för hur olika funktioner i återställningsverktyg ser ut i verkligheten. Kopplingen mellan fordon och en dator kan göras via en USB nyckel och en VCI det vill säga Vehicle Communication Interface, som är ett gränssnitt för att kommunicera med ett fordon och fungerar som en mellanhand mellan en dator och fordon. Efter koppling av fordon med datorn är klar kan återställningsverktyget börjas på datorn. Pilarna i bilden hur kommunikationen är tänkt och hur olika spår beter sig. Alla fyra spår behöver anslutning till fordon för att utföra sina funktioner. Återställningsfunktion baseras på backup-mappen och ny baseline funktion baseras på delar av backup mappen. Mer detaljer om olika funktioner finns beskrivna nedan.

4.3.3 Backup

Backupflödet har funktioner som sparar fordonets eller riggens tillstånd för att återställa fordonet eller riggen efteråt. SOPS-filen ska laddas in från fordonet för att få reda på fordonets identifieringsnummer, s.k. VIN. Efter läsning av SOPS-filen kan man börja med att skapa en mapp för att spara all data. Mappen kan namnges med hjälp av fordonets identifieringsnummer tillsammans med datum och tid när sparandet gjordes av användaren. Efter att mappen är skapad skrivs SOPS-filen i krypterad form i mappen.

Därefter ska all nödvändig information sparas från varje styrenhet. Här ingår bland annat ett komplettnummer bland ECU data, som visar vilken mjukvaruversion finns installerad och aktuella parametrar på alla enheter. Dessutom sparas en så kallad demofil i samma mapp där resten av information sparades. Som ett extra stöd för användarna bör varje enskild fil innehålla fordonets identifieringsnummer i namnet. Hela backupflödet loggas till den nyligen skapade backup mappen som innehåller viktiga detaljer som aktiviteter med tidsstämpel och status på hur aktiviteter gick vid processen. Dessutom innehåller det även användarnamn och maskinnamn som användes för att göra en backup. Även totala tiden för hela processen sparas.

4.3.4 Återställning

Återställningen är processen som skriver tillbaka de tidigare sparade värden till fordonet eller riggen. Först ska fordonets identifieringsnummer, det vill säga, VIN tas fram för att sedan kunna välja rätt mapp med återställningsdata. På det sättet kan man försäkra att återställningsprocessen utförs på rätt fordon genom att matcha identifieringsnummer. I nästa steg ska styrenheternas komplettnummer läsas och jämföras med de sparade värdena från backup mappen för att avgöra om enheten behöver flashprogrammeras.

När jämförelsen är klar ska flashningen ske på enheterna som behöver det. Flashprogrammering måste ske innan SOPS-filen och parametrarna skrivs tillbaka till fordonet. Detta för att styrenheterna kan tappa sina data när de flashprogrammeras. Efter skrivning av SOPS-filen raderas även felkoder i styrenheterna som följer med nollställning av styrenheter. Hela återställningsflödet loggas till den backup mappen där återställning utgick ifrån. Loggfilen innehåller viktiga detaljer som aktiviteter med tidsstämpel och status på hur aktiviteter gick vid processen. Dessutom innehåller det även användarnamn, maskinnamn som användes för att göra en återställning. Även totala tiden för hela återställningsprocessen sparas.

4.3.5 Reservdelsprogrammering

Den här funktionen är ett annat sätt att parametersätta olika styrenheterna som finns i ett fordon eller en rigg. Skillnaden är att man skriver parametrar utifrån SOPS-filen. För att göra det och som i de andra funktionerna måste SOPS-filen läsas först från fordonet. SOPS-läsning fungerar i princip på samma sätt som den gjordes vid backup- och återställningsspår. Med hjälp av SOPS-filen och en databasfil sätts sedan parametrar av alla styrenheter enligt databasen tillbaka till fordonet.

Efter reservdelsprogrammering är klar måste SOPS-filen skrivas tillbaka till fordonet. Det här skrivandet av SOPS-filen beror på att reservdelsprogrammeringen gör vissa ändringar i den också. Efter det raderas alla felkoder från styrenheterna och en nollställning görs på dem.

Precis som andra funktioner loggas även den här processen. Loggfilen innehåller viktiga detaljer som aktiviteter med tidsstämpel och status på hur aktiviteter gick vid processen. Dessutom innehåller det även användarnamn, maskinnamn som användes för att göra reservdelsprogrammering. Även totala tiden för hela processen sparas. En skillnad vid loggfil sparandet är att det läggs under samma mapp som verktyget ligger i en separat loggmapp för att den inte är kopplad till en backup mapp.

4.3.6 Ny baseline

Denna funktion inkluderades i återställningsverktyg i efterhand av uppdragsgivaren för att öka flexibiliteten av slutprodukten ur företagets perspektiv. Det går ut på att användaren får möjlighet att sätta upp fordonet enligt en ny baseline. Baseline innebär ett tillstånd som för den här funktionen betyder att mjukvaruversioner av styrenheterna kan sättas oavsett vilket tillstånd fordonet hade innan. Idén för att sätta en ny baseline kan relateras igen till flashprogrammering där man kan ändra mjukvaruversioner utifrån komplettnummer på olika styrenheter.

För att fullborda den här funktionen väljer användaren en lista med komplettnummer av olika styrenheter som anpassas till önskade mjukvaruversioner och en sparad SOPS-fil. Därefter flashprogrammerar verktyget alla styrenheter till den önskade versionen som finns med i listan. Som nämns i återställningsprocessen måste man ta hänsyn till skrivandet av SOPS-filen och reservdelsprogrammering igen efter flashning. Detta för att styrenheterna kan tappa sina data när de flashprogrammeras. Då görs en reservdelsprogrammering för alla enheter utifrån den valda SOPS-filen. Därefter skrivs den valda SOPS-filen tillbaka till lämpliga styrenheter.

Som andra funktioner loggas även den här processen. Loggfilen innehåller viktiga detaljer som aktiviteter med tidsstämpel och status på hur aktiviteter gick vid processen. Dessutom innehåller det även användarnamn, maskinnamn som användes för att göra reservdelsprogrammering. Även totala tiden för hela processen sparas. Den här loggfilen sparas i samma backup mapp som valdes i början för att välja en SOPS-fil och fil med lista av önskade komplettnummer.

4.4 Installation

Verktyget installeras från en exekverbar fil med filnamn extension i .exe format. Installationsfilen skapas med hjälp av programmet InnoSetup. Det går till genom att komponenterna hämtas från Scanias release-server där de interna applikationerna och biblioteken publiceras. Utifrån de interna delarna samt de nödvändiga filerna som utvecklingsmiljön Visual Studio bygger av källkoden skapar InnoSetup verktyget .exe filen som installerar alla filer nödvändiga för fungerandet av återställningsverktyget.

4.5 Uppdateringar

Det implementerades inget riktigt uppdateringssystem med automatisk kontroll i slutprodukten. Även nedladdning och installation av de uppdateringarna avveks på grund av brist av utvecklingstiden. När komponenterna MainLibrary och Flasher uppdateras med sitt innehåll måste de nya ändringarna installeras på återställningsverktyg genom att skapa en ny installerare. Det tas hand av personen som kommer att vara ansvarig för förvaltning av återställningsverktyg på företaget. För att DevelopmentTool har sin egen separat installationen sköts uppdateringar från det automatiskt och det behöver inte ingå i uppdateringsprocessen.

4.6 Testning

Prototypen har provats kontinuerligt under utvecklingsarbetet på olika testriggar. Efter slutprodukten var färdigutvecklad testades den på både testriggar och fordon. Testningen handlade mest om att testa alla fyra funktioner som slutprodukten har. Då gällde det att följa om flödet för de fyra funktionerna följs som de ska. Efter det fick hela avdelningen använda det under en testomgång, som innebär en vecka av dagligt testningsarbete på fordon. Här har prototypen använts tillsammans med den manuella processen för att jämföra och säkerställa att återställningsverktyget utför allt som det ska, samma saker som ingår i den manuella processen. De detaljerade testerna finns beskrivna i bilaga 4.

Testerna jämfördes med användningsfallen så att funktionerna utförs enligt dem. Olika testfall har visat att prototypen fungerar som förväntat och uppfyller alla delmål. Prototypen kommer att användas på företaget som tänkt, det vill säga att den kommer att ersätta den manuella processen. Felen som uppstod under testningsvecka av utvecklarteamet åtgärdades direkt på plats så att prototypen kan fungera som det ska.

5 Analys och diskussion

En prototyp har tagits fram för att kunna avgöra om rätt val har gjorts för återställningsverktyg under förstudien. Slutprodukten är uppbyggd av ett bibliotek som innehåller logiken och två gränssnitt. Det ena gränssnittet är grafiskt som är uppbyggd med hjälp av Windows Forms i Visual Studio. Det utvecklades för en förenklad användning av produktens funktionaliteter. Det andra är ett kommandoradsgränssnitt som möjligen kan användas för återanvändning i framtiden inom företaget. I detta kapitel beskrivs detaljerna av prototypen mer ingående.

Projektets konsekvenser för samhället ur ekonomisk, social, etisk och miljömässig synvinkel diskuteras som följande. Tillämpning av integrationsmodellen för att utveckla en slutprototyp gör testningsarbete på företaget enklare så att tiden kan läggas till någon användbar aktivitet. Det är konsekvens för samhället ur ekonomisk och social synvinkel så att företaget kan använda sina resurser mer effektivt. Återställningsprocessen förbättrar arbetsmiljö för att den är ett långt och tråkigt arbete som inte behöver göras. Projektet har inga direkta konsekvenser för miljön.

5.1 Integrationen

Integrationen på användargränssnitt- och applikationsgränssnittnivå enligt EAI i slutprodukten gör att produktutvecklingen gick snabbare genom att använda de befintliga resurserna på företaget än de skulle återutvecklas. Utvecklingsprocessen gick effektivt för att ägna tid åt nya funktioner som byggs på integrerade applikationer. Utöver de fördelarna kan man också konstatera att produkten resulterades till ett långsiktigt åtagande som kräver att ha kontroll på saker som uppdateringar av delapplikationerna eller installation av de uppdaterade versionerna i slutprodukten.

Anledning för att använda applikationsprogrammeringsgränssnittnivå (API) för SOPS-hantering och flashprogrammering är att få bättre kontroll av hur olika metoder implementeras och enklare fel hantering i systemprototypen. Mer konkreta fördelar för varje funktion tas upp i avsnitt 5.3. När det gäller användargränssnittnivå är den största fördelen att implementering blir väldigt enkelt med mindre kod och fel som gör att byta komponenten blir enklare. Mer konkreta nackdelar och fördelar för systemprototyp beskrivs under avsnitt 5.3.2.

5.2 Modularitet

Prototypen byggdes med fokus på modularitet. Detta innebär att de ovanstående komponenterna implementeras på ett sätt som gör det möjligt att enkelt byta ut dem. I första hand innebär det att principerna inom objektorienterade programmering följs så att det finns en tydlig uppdelning av flödet. När det gäller uppdelning av olika processer är slutprodukten uppbyggd så att varje metod gör ett välavgränsat arbete.

Genom att tillämpa modularitet i återställningsverktyg är det enkelt att byta implementationen, till exempel genom att byta vilken komponent utför arbetet, utan att resten av applikationen påverkas. Strukturen av prototypen är byggd så att logiken är helt skild från själva gränssnittet vilket gör att implementationen av ett grafiskt gränssnitt och ett kommandoradsgränssnitt gjordes väldigt enkelt utan att ändra i modellen. Slutprodukten har möjlighet för utbytbarhet av integrerande applikationer om det behövs framöver.

5.3 De integrerade komponenterna

Här beskrivs vilka komponenter har valts för integrationen för slutprototypen och anledning för gjorda val bland de applikationerna som var tillgängliga för att integreras som också diskuterades i avsnitt 2.4, Undersökta verktyg för integration.

5.3.1 Hantering av SOPS-filen

Hantering av SOPS-filen görs via MainLibrary. Att det redan krävs av andra komponenter betyder att det kan återanvändas och slutprodukten behöver inga nya beroenden. Applikationsprogrammeringsgränssnittnivå (API-nivå) integrationen innebär även bättre kontroll över processen som var fördelaktigt för att kunna uppfylla kraven som ställdes på prototypen av företaget.

5.3.2 ECU-parametrar

DevelopmentTool används för att hantera parametrarna i styrenheterna. Eftersom verktyget DevelopmentTool används på användargränssnittnivå blev implementering väldigt enkelt. Nackdelen med användargränssnittnivå integration för slutprototypen betyder också att uppdateringen av DevelopmentTool kommer att bli komplicerad. På grund av användning av detta verktyg kunde man återanvända metoderna som fanns med i applikationen redan med hjälp av kommandoradsgränssnitt och bygga upp nya funktioner ovanpå dem.

5.3.3 Annat ECU data

För hantering av annat ECU data som enheternas namn, CAN-adress och unikt komplettnummer valdes MainLibrary. Detta bibliotek förenklade utvecklingsprocessen för att den hade en detaljerad API:n för att utgå ifrån. Det möjliggjorde även att processflödet som skapades för återställningsverktyget kunde styras på grund av API-nivå. Styrningen av flödet blev enkelt genom att spara bara det nödvändiga datat från styrenheterna i enkelt hanterbart format i slutprodukten. Det datat kunde vidare användas för flashning och reservdelsprogrammering.

5.3.4 Hantering av felkoder

I prototypen används MainLibrary för att hantera felkoder från styrenheterna. Detta på grund av att DevelopmentTool, som annars ansågs vara ett bättre alternativ för uppgiften stödjer inte funktionen från sitt kommandoradsgränssnitt. Den enda svårigheten med användning av MainLibrary var att användaren inte får någon beskrivning av felkoder i styrenheter, som är viktigt om användaren vill läsa av felkoden och vill veta mer om hur de felkoderna kan diagnostiseras.

5.3.5 Flashprogrammering

Flasher valdes för att utföra flashprogrammering. Integrationen av Flasher i slutprodukten var på API-nivå integrationen i EAI-modellen. Flasher hade ingen avgörande fördel utan snarare att de andra alternativen hade för stora nackdelar. Själva integrationsprocessen gick enkelt vid utveckling av slutprodukten efter en bra demo på vilka funktioner som anropas vid flashning i Flasher.

5.3.6 Reservdelsprogrammering

MainLibrary har också valts för reservdelsprogrammering återigen för att det måste implementeras som en del av andra komponenter, så det satt inga krav nya beroenden i slutprodukten. MainLibrary för reservdelsprogrammering var relativt enkelt att implementera och var väldigt bra förklarat i API:n. Eftersom även SOPS-läsning i återställningsverktyg från fordon görs via detta bibliotek var det smidigare att bygga på samma applikation än de andra alternativen.

5.4 Författarnas bidrag

Författarna var lika insatta i uppgiften och utveckling av slutprodukten. Alla dokument som krävdes underhållning togs hand om av författarna ständigt under examensarbete. Skriftlig dokumentation av återställningsverktyg i form av en rapport skrevs av båda författarna. För det första introducerades författarna till vilka verktyg som möjligen kan användas under integrationsprocessen av handledaren på Scania och vilka borde intervjuas angående de olika verktygen. Även första utkastet av återställningsprocessen togs fram av handledaren för att hjälpa författarna att komma igång med examensarbete.

6 Slutsatser

En utvärdering av prototypen och hur integrationen tänkts för utveckling av återställningsverktyg tas upp under det här kapitlet. Dessutom avslutar kapitlet med några framtida möjligheter för vidareutveckling av slutprototypen.

6.1 Prototypen

Automatisering av en befintlig process som utförs manuell är alltid givande och bidragande både för företagets utveckling och för effektivisering av resursanvändning. Återställningsprocessen var ett tidskrävande jobb för testteamet på företaget och genom att integrera applikationer för att återanvända resurser kan automatiseringen betyda ett stort framsteg. Uppgiften förutom utveckling av återställningsverktyg med givna krav var också att utföra integrationen på ett så effektivt sätt som möjligt. Med möjligheter på hur slutprodukten kommer att användas och förvaltas på företaget ingick uppgiften att hålla integrationen av applikationer på ett sätt så att beroendet kan hållas relativt lågt.

Uppgiften gick också ut på att ta reda på de applikationerna som möjligen kan användas för integrationen och att fatta beslut om hur den slutliga integrationen ska se ut med relevanta förutsättningspunkter. Resonemanget kring vilka nivåer ska applikationer hållas till enligt EAI integrationsmodellen var en stor del av examensarbetet. Uppbyggnaden av slutprodukten baseras på koncept av modularitet så att vid pågående utveckling av applikationer på företaget kan vissa moduler i återställningsverktyg bytas ut smidigt och utan att påverka andra funktioner direkt. Vilka tankesätt som låg bakom de valda integrationslösningarna för återställningsverktyg vara ett bidrag till att förenkla val bland integrerande applikationer.

Prototypen blev till slut en integrerad applikation som baserades på EAI-modellen och använder tre komponenter på olika nivåer, nämligen applikationsprogrammeringsgränssnitt och användargränssnitt nivå. Själva prototypen är en Windowsform applikation utvecklades i programmeringsspråket C# som kan utföra fyra huvudfunktioner. De funktionerna är sparandet av tillstånd av fordon, återställning av fordon med hjälp av sparade tillståndet, reservdelsprogrammering och tillämpning av en förutbestämd baseline för fordon..

6.2 Framtida utvecklingen

Återställningsverktyg till slut tar upp fyra huvudfunktioner med modularitet och EAI-modellen som stödjepunkt. Men det finns några möjligheter eller förbättringsförslag som kan läggas till i slutprodukten för att göra det ännu mer stabilt, effektivt och återanvändbart i framtiden. Den första möjligheten är att uppdateringen kan göras inbyggd och automatisk från slutprodukten än att manuellt behöver tänka på den. Efter en automatisk uppdateringskontroll kommer slutprodukten att bli mer dynamisk i användningen.

Vid implementeringen av DevelopmentTool har produkten inget sätt att kontrollera om det har exekverat sin funktionalitet via kommandoradsgränssnitt rätt eller inte. Denna funktion skulle göra återställningsverktyg ännu mer pålitlig vid integration. Det kom en efterfråga om enhetstestning för slutprodukten i mitten av examensarbetet för att förenkla förvaltningsprocessen för företaget framöver. Enhetstestning kunde inte prioriteras på grund av tidsbrist och författarnas brist på kunskaper om enhetstestning av moduler och metoder speciellt i den valda utvecklingsmiljön.

Kalibreringsvärden skulle också kunna implementeras som ett förbättringsförslag i återställningsflödet. Efter utforskning av relevant information som vilka styrenheter har denna typ av värde, vilka verktyg som är tillgängliga för att läsa och skriva värde samt hur de skulle kunna implementeras i slutprodukten.

Produkten i dagsläget stödjer inte sparandet och skrivning av parametrar för alla styrenheter i fordon. Stödet finns bara för de styrenheterna som DevelopmentTool har stöd för. Efter diskussion med uppdragsgivaren och teamet har det kommits fram till att utvecklarna av DevelopmentTool kommer att få kravställning internt för att få stöd för de icke-stödda styrenheterna. Det för det första att minska komplexitet för återställningsverktyg som gör att man kan undvika att blanda in nya applikationer för att fullborda den funktionen i integrerade slutprodukten. Stöd för att spara adaptionsdata som finns i DevelopmentTool men inte kommandoradsgränssnitt kommer att läggas vid senare tillfälle.

7 Källförteckning

[1] Scanias webbsida, Huvudrubrik: Om Scania, Underrubrik: *Scania i Sverige*, Källa: http://www.scania.se/om-scania/scania-i-sverige/, Hämtad: 2016-03-17.

[2] Junyoul lee, Keng Siau, Soongoo Hong, *Enterprise Integration with ERP and EAI*, Communications of the ACM, Upplaga: February 2003, Volym: 46, Nummer 2, Hämtad: 2016-05-14.

[3] Goldstone Technologies Limited, *Enterprise Application Integration - an overview*, White paper, Källa: http://www.goldstonetech.com/investor%20info/white%20papers/EAI%20Overview.pdf, Hämtad: 2016-05-15.

[4] Ananias Laftsidis, *Enterprise Application Integration*, Kapitel 5, IBM Sweden, Linthicum, 2000, Källa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.200.6603&rep=rep1&type=pdf Hämtad: 2016-04-26.

[5] Alon Y. Halevy, Naveen Ashish m. fl, *Enterprise Application Integration: Successes, Challegners and Controversioes*, University of Washington, Nasa, Microsoft m fl., Källa: https://homes.cs.washington.edu/~alon/files/eiisigmod05.pdf, Hämtad: 2016-04-28.

[6] Tariq Rahim Soomro, Abrar Hasnain Awan; *Challenges and Future of Enterprise Application Integration*, International Journal of Computer Applications, Department of Information Technology, SZABIST Dubai Campus, UAE, 2012, Källa: http://research.ijcaonline.org/volume42/number7/pxc3877762.pdf, Hämtad: 2016-03-16.

[7] Ariyan Fazlollahi, *Benefits of Enterprise Integration Systems*, Master thesis, Industrial Information and Control Systems, Department of Electrical Engineering, Kungliga Tekniska Högskolan, Stockholm, Sweden, 2012. Källa: http://kth.diva-portal.org/smash/get/diva2:537365/FULLTEXT01.pdf, Hämtad: 2016-03-16.

[8] Rikard Land, Ivica Crnkovic; *Existing Approaches to Software Integration – and a Challenge for the Future*, Workshop paper, 2004, Department of Computer Science and Engineering, Mälardalen University, Västerås, Sweden, Källa: http://www.es.mdh.se/pdf_publications/642.pdf, Hämtad: 2016-03-22.

[9] Rikard Land, *An Architectural Approach to Software Evolution and Integration*, Licentiate Thesis, 2003, Department of Computer Science and Engineering, Mälardalen University, Västerås, Sweden. Källa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.7586&rep=rep1&type=pdf, Sida: 97-108, Hämtad: 2016-03-23.

[10] Gabriella Björk, *Värdeskapande genom modularisering, Kandidatexamensarbete* i Integrerad Produktutveckling, KTH Industriell teknik och management, Kungliga Tekniska Högskolan, Stockholm, Sweden, 2015, Källa: http://kth.diva-portal.org/smash/get/diva2:895624/FULLTEXT01.pdf, Hämtad: 2016-03-23.

[11] Scania Value, Moduler Modellen för framgång, *Framgång genom modularisering*, Tema: Modulsystem, Kvartal: 4, År: 2009, QX/200X, Källa: http://se.scania.com/images/SV_72_tcm120-181094.pdf, Hämtad: 2016-05-07

8 Bilagor

Bilaga 1, Intervjuer

Följande frågor ställdes till de olika utvecklargrupperna för applikationerna som undersöktes under förstudien.

- Beskrivning av applikationer från respektive utvecklingsgrupper blev en självklar fråga för att säkerställa att all information har tagits hänsyn till innan utveckling av slutprodukten skulle börjas.
- Information om vilka delar som möjligen kunde användas för återställningsverktyg ingick också här. Med delar menas det olika komponenter, moduler eller applikationer som eventuellt stödjer integration.
- Frågor som tog upp vilka funktioner i applikationen kan läggas till återställningsverktyg så att integrationen kan bli så effektiv som möjligt med så få integrerade komponenter som möjligt för att minska beroendet i slutprodukten.
- Om integrationen är möjligt, på vilka sätt kan det utföras i återställningsflödet. Om de inte stödjer integrationen, finns det alternativa vägar att välja för att åstadkomma till funktioner på ett annat sätt.
- En fortsättningsfråga om integration är möjligt, vilken nivå tyckts vara bäst så att förvaltningen av produkten blir så enkel som möjligt.
- Övrig relevant information som kan vara bra att veta ifall applikationen påverkas av andra integrerade applikationer vid utveckling. Påverkan kan vara på olika nivåer, exempelvis om två applikationer använder samma komponent i olika versioner, vilken version ska föredras så att utvecklingen kan fortsätta utan någon störning.
- Alternativa förslag för de funktionerna återställningsverktyget behöver ingick som en relevant del i intervjuer. Vissa av de hänger direkt ihop med utveckling och vissa så att den framtida efterfrågan kan underlättas vid vidareutveckling.
- Till vissa avdelningar som jobbar med liknande uppgifter ingick frågor som tog upp om något liknande som återställningsverktyg har gjorts innan på företaget i intervjun. Om svaret är ja, kunde frågorna som vad de är, hur man kan få tag på dem och om det finns delar av dem som kan vidareutvecklas eller återanvändas väcktes upp.

Bilaga 2, Användningsfall

A. Användningsfall 1

Skapa en backup

Aktörer:

- Användaren
- Fordonet

Förutsättningar:

- Datorn är kopplad till fordonet via VCI
- USB nyckeln sitter i datorn

Normalt flöde:

1: Användaren trycker på *Connect* knappen
2: Uppkoppling skapas mot alla styrenheter
3: Knapparna blir aktiva
4: Användaren trycker på *Backup* knappen
5: Användaren får välja en mapp där informationen ska sparas
6: SOPS-filen läses från fordonet för att få ut fordonets identifierings nr
7: Mappen för datat skapas med identifieringsnumret, datumet och tiden
8: SOPS-filen sparas i mappen
9: ECU data sparas i en textfil
10: DevelopmentTool startas för att spara parametrar från styrenheter
11: Demofilen sparas vid anrop till DevelopmentTool
12: Felkoder sparas
13: Loggfilen för backup flödet sparas

B. Användningsfall 2

Återställa från en backup

Aktörer:

- Användaren
- Fordonet

Förutsättningar:

- Datorn är kopplad till fordonet via VCI
- USB nyckeln sitter i datorn

Normalt flöde:

1: Användaren trycker på *Connect* knappen

2: Uppkoppling skapas mot alla styrenheter

3: Knapparna blir aktiva

4: Användaren trycker på *Restore* knappen

5: SOPS-filen läses från fordonet för att få ut fordonets identifierings nr

6: Användaren får välja backup mappen som den vill återställningen ska hämta data från

7: ECU data hämtas från fordonet

8: ECU datat jämförs med det sparade för att avgöra vilka enheter behöver flashas

9: De styrenheterna som behöver det flashprogrammeras

10: SOPS-filen skrivs tillbaka till fordonet

11: De styrenheterna som blev flashade och är inte stödda av Development-Tool reservdelsprogrammeras

12: Sparade parametrar skrivs tillbaka av DevelopmentTool

13: Felkoder tas bort

14: Styrenheterna nollställs.

15: Återställning loggfilen sparas i en annan mapp i backup mappen

Alternativa flöden:

6a: Användaren väljer mapp som inte stämmer med fordonets identifierings nr

6a i: Användaren får ett felmeddelande och programmet går till steg 6

C. Användningsfall 3

Reservdelsprogrammering

Aktörer:

- Användaren
- Fordonet

Förutsättningar:

- Datorn är kopplad till fordonet via VCI
- USB nyckeln sitter i datorn

Normalt flöde:

1: Användaren trycker på *Connect* knappen

2: Uppkoppling skapas mot alla styrenheter

3: Knapparna blir aktiva

4: Användaren trycker på *Spare part program* knappen

5: SOPS-filen läses från fordonet för användning i processen

6: Alla styrenheter reservdelsprogrammeras med hjälp av en databas och SOPS-filen

7: SOPS-filen skrivs tillbaka till fordonet

8: Loggfilen sparas i slutproduktens mapp

D. Användningsfall 4

Ny baseline

Aktörer:

- Användaren
- Fordonet

Förutsättningar:

- Datorn är kopplad till fordonet via VCI
- USB nyckeln sitter i datorn

Normalt flöde:

1: Användaren trycker på *Connect* knappen

2: Uppkoppling skapas mot alla styrenheter

3: Knapparna blir aktiva

4: Användaren trycker på *New Baseline* knappen

5: Användaren får välja en SOPS fil

6: Användaren får välja en textfil som innehåller listan av önskade komplettnummer

7: Alla styrenheter flashprogrammeras till önskad komplettnummer

8: Alla styrenheter reservdelsprogrammeras med hjälp av en databas och SOPS-filen

9: SOPS-filen skrivs till fordonet

10: Baseline loggfilen sparas i mappen

Alternativa flöden:

7a: Innehållet i textfilen stämmer inte

7a i: Flödet avbröts

7b: En önskad komplettnummer är inte möjlig för den styrenhetshårdvaran

7b i: Processen fortsätter med nästa styrenhet, avvikelsen loggas.

Bilaga 3, Verktyg tillgängliga för integrationen

- **SOPS-filen**

	SOPS-läsning	SOPS-skrivning
SOPSHandler	x	x
MainLibrary	x	x
MainLibraryGUI	x	x
ECUtool	x	x

- **ECU data**

	ECU namn	ECU komplett-nummer	ECU CAN-adresser
MainLibrary	x	x	x
DevelopmentTool	x	x	
MainLibraryGUI	x	x	x

- **ECU felkoder**

	DTC läsning	DTC radering
MainLibrary	x	x
DevelopmentTool	x	x
MainLibraryGUI	x	x

- **ECU parametrar**

	Parameter läsning	Parameter skrivning
DevelopmentTool*	x	x
MainLibraryGUI	x	

* stödjer inte alla enheter

- **Flashprogrammering och reservdelsprogrammering**

	Flashprogrammering	Reservdelsprogrammering
MainLibrary		x
Flasher	x	
FlashLibrary	x	
SOPSHandler		x
DevelopmentTool	x	
MainLibraryGUI		x
ECUtool	x	x

Bilaga 4, Testning

A. Backup

För att testa sparandet av fordontillstånd följdes backup-flödet via de här stegen,

- Efter anslutning till fordon, det vill säga när alla styrenheter svarar, kan man börja med testning av sparandet.
- Testning av om SOPS-filens läsning går som förväntat från de specifika enheterna som innehåller SOPS-filen. Om processflödet går rätt, loggas den med tidsstämpel och från vilken enhet filen lästes ifrån.
- Testa om läsning av identifieringsnummer av fordonet från SOPS-filen har gott rätt. Om det gjordes som förväntas kan testas om numret visas korrekt på gränssnittet och loggas rätt på slutet av backup loggfilen.
- Att rätt mapp skapades testades och om det finns på rätt filsökväg och med rätt format på mappnamn. Identifieringsnummer och datum- och tidsstämpel bör stämma i mappnamnet.
- Sparandet av SOPS-filen testades efter kommunikationen med fordonet och om filen sparades i ett krypterat format med rätt identifieringsnummer.
- Efter läsning av SOPS-filen testades det om läsning av komplettnummer av styrenheterna stämde med hjälp andra applikationer som kan läsa numret. Samma testfall gäller för läsning av CAN-adress, enhetsnamn och felkoder.
- Eftersom information som komplettnummer, CAN-adress och namn av alla enheterna i fordon sparas i en annan fil, ingick steget för att testa om sparandet av filen gick bra också med rätt identifieringsnummer av fordon i filnamn.
- Loggning av om alla steg bland annat svar från olika enheter vid anslutning och läsning av felkoder för enheterna går som förväntat testades. Processen skulle fortsätta även när fel uppstod.
- Utöver det ingår testning av sparandet av parametrar i styrenhet med hjälp av DevelopmentTool med rätt format och i rätt mapp i backup-mappen. Testning om demo-filen sparades rätt och i rätt format. Även här gäller loggningstest om vilka enheter sparas och hur det gick till.
- Testning tog också upp om rätt användare, rätt maskinnamn och hur långt det tog för att utföra hela backup-processen på datorn där backup gjordes ifrån loggas i backup loggfilen.

Utfall

Tester utfördes stegvis och rättades omgående om felen uppstod.

B. Återställning

För att göra återställning av fordontillstånd följdes återställningsflödet via de här stegen,

- Efter anslutning till fordon, det vill säga när alla styrenheter svarar, kan man börja med testning av återställningsflödet som är en av de kärnfunktionerna i slutprodukten.
- Testning av om SOPS-filens läsning av fordon går som förväntat från de specifika enheterna som innehåller SOPS-filen. Om processflödet går rätt, loggas den med tidsstämpel och från vilken enhet filen lästes ifrån. Då kan resten av processen fortsätta.
- Den första delen i återställning börjar med att testa om mappen som är vald är rätt för att göra backup med. Det görs enklast om identifieringsnummer av fordon matchar med identifieringsnummer av fordon i den valda backup mappen. Det testas i verktyget under körningen då om det är fel kan man inte gå vidare med återställningsprocessen tills rätt backup mapp väljs.
- Andra delen är att kontrollera om rätt backup mapp valts. SOPS-fil läsning från sparade mappen testades så att skrivning på slutet av processen kan göras till rätt enheten.
- Testning av att jämföra komplettnummer av alla styrenheter sparade i backup mappen görs med komplettnummer från enheterna i fordon för att lista ut vilka som ska flashas.
- Flashningen testades också genom att ansluta fordon till något annat verktyg som visar komplettnummer efter flashning. Då kan man säkerställa att flashning har fungerat som det ska. Annars loggas resultat vid felmeddelande i återställning loggfilen med aktivitet och vilken styrenhet som fick felmeddelandet.
- Ifall skrivning av styrenheter inte stöds av DevelopmentTool måste de parametersättas med hjälp av SOPS-filen. Det testades genom att följa om reservdelsprogrammering utfördes på de styrenheterna utifrån SOPS-filen. Därmed testades om loggningen av detta steg görs rätt med rätt tidsstämpel och med rätt lista av enheter med rätt status.
- Skrivandet av SOPS-filen testades också efter flashningssteget samt med tillhörande loggning i återställning loggfilen.
- DevelopmentTool används för att skriva parametrar som det har stöd för. Detta steg testades bara med hjälp av utskriften som man får via kommandoradsgränssnitt. Det här är egentligen det säkraste testet som man kan göra men det är en del av förbättringsmöjlighet för produkten. Loggning av vilka styrenheter som parametersättas av DevelopmentTool testades också. Lista bör stämma med enheter som finns med i backup mappen som skapades också via DevelopmentTool vid sparandet med aktiviteten och status med tidsstämpel.

- Vid slutet av varje flöde testas om radering av felkoder från alla styrenheterna går rätt som följs av nollställningen. Även här ingick testning om loggning av de stegen via felkodsraderingen och nollställningen gjordes rätt.
- Det kontrolleras också om användar- och datornamn från operativsystemet, samt total tid och mjukvaruversioner hamnar rätt i loggfilen.

Utfall

Tester utfördes stegvis och rättades omgående om felen uppstod.

C. Reservdelsprogrammering

För att testa reservdelsprogrammering av fordontillstånd följdes flödet via de här stegen,

- Efter anslutning till fordon, det vill säga när alla styrenheter svarar, kan man börja med testning av reservdelsprogrammering.
- Testning av om SOPS-filens läsning går som förväntat från de specifika enheterna som innehåller SOPS-filen. Om processflödet går rätt, loggas den med tidsstämpel och från vilken enhet filen lästes ifrån.
- Testa om läsning av identifieringsnummer av fordonet från SOPS-filen har gott rätt. Om det gjordes som förväntas kan testas om numret visas korrekt på gränssnittet och loggas rätt på slutet av reservdelsprogrammering loggfilen.
- Näst i flödet blir testning av om alla enheter kan göra reservdelsprogrammering utifrån SOPS-filen. Resultatet av programmeringen sparas i loggfilen med tidsstämpel och vilka enheter gjorde programmering och respektive status meddelande om hur det gick. Vid fel skrivs felmeddelandet i filen och att processen fortsatte för andra styrenheter utan att avbryta flödet testades.
- Efter testning av flashning av styrenheterna ingår ett steg att skriva tillbaka SOPS-filen till tillhörande styrenheterna som är tillsatta för att innehålla filen. Även för detta steg testas om loggning i reservdelsprogrammering loggfilen går rätt med rätt identifieringsnummer och korrekta status från aktiviteter.
- Vid slutet av varje flöde testas om radering av felkoder från alla styrenheterna går rätt som följs av nollställningen. Även här ingick testning om loggning av de stegen via felkodsraderingen och nollställningen gjordes rätt.
- Det kontrolleras också om användar- och datornamn från operativsystemet, samt total tid och mjukvaruversioner hamnar rätt i loggfilen.

Utfall

Tester utfördes stegvis och rättades omgående om felen uppstod.

D. Ny baseline

För att testa ny baseline spåret av återställningsverktyg, gjordes följande tester,

- Efter anslutning till fordon, det vill säga när alla styrenheter svarar, kan man börja med testning av om ny baseline har satt till fordonet eller ej.
- Flashningen av styrenheterna testas på samma sätt som det gjordes för flashningen i återställningsflödet.
- Reservdelsprogrammering och skrivning av SOPS-filen tillbaka testas också på samma sätt som det gjordes i återställningssteg.
- Vid slutet av varje flöde testas om radering av felkoder från alla styrenheterna går rätt som följs av nollställningen. Även här ingick testning om loggning av de stegen via felkodsraderingen och nollställningen gjordes rätt.
- Det kontrolleras också om användar- och datornamn från operativsystemet, samt total tid och mjukvaruversioner hamnar rätt i loggfilen.

Utfall

Tester utfördes stegvis och rättades omgående om felen uppstod.

TRITA 2016:61