Emrik (EC)

Skatteverkets anslutningstjänst till Emma/Mina meddelanden

Software Architecture Document

(Arkitekturdokument)

Version 0.2

Revision: 734

Mall: software_architecture_document.dot
Mallversion: 2.5
Mallansvarig: SM-uppdraget

Verksamhetsstöd för skatt och KFM

Sammanfattning

Denna arkitekturbeskrivning av systemet Emrik har tagits fram efter systemet tagits i drift som en dokumentation av systemets utförande och egenskaper.

Emrik är en anslutningstjänst som är tänkt att underlätta för verksamhetssystem hos Skatteverket att ansluta sig till ett system för digital meddelandeförmedling (MFS), i praktiken innebär detta anslutning till systemet Emma, det enda MFS som existerar idag, och som är utvecklat och driftas av Skatteverket.

Innehållsförteckning

1. Inledning

1.1 Syfte

Detta dokument innehåller en översikt av systemet ”Emrik”, med hjälp av ett antal arkitektoniska vyer beskrivs olika aspekter av systemet för olika intressenter. Det är ämnat att fånga och framföra de signifikanta arkitektoniska beslut som gjorts om systemet.

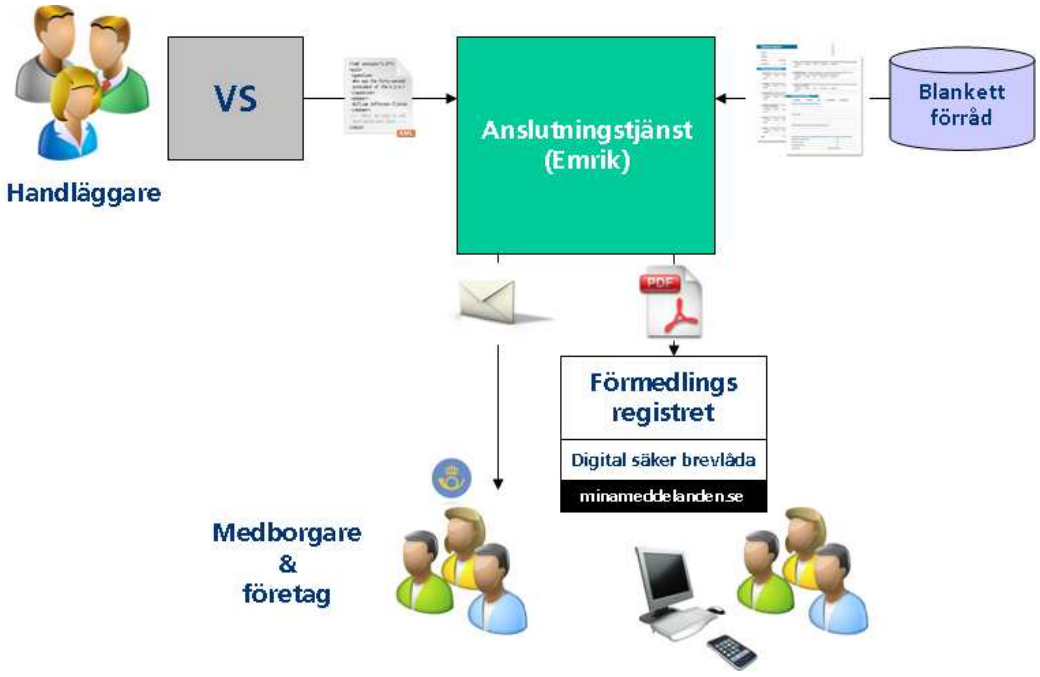
1.2 Målgrupp

Arkitekturdokumentet riktar sig till alla inblandade i utvecklingen/underhållet av *Emrik*. Vissa delar är av generell karaktär andra riktar sig främst till arkitekter och utvecklare. Därmed omfattas såväl beställare som utförare av infrastruktur och verksamhetssystem.

1.3 Bakgrund

MinaMeddelanden är en myndighetsgemensam tjänst som byggts för att göra det möjligt för myndigheter och kommuner att ersätta utskick av papperspost med digitala meddelanden till medborgare och företag på ett säkert sätt.

Emrik är ett system som byggts på Skatteverket för att förenkla integrationen av verksamhetssystem och MinaMeddelanden på Skatteverket. Emrik ligger mellan verksamhetssystemet (via ZI’s utskriftstjänster) och RPH och delar upp utskriftsflödet mellan pappers- och digitala utskick. Om mottagaren har registrerat sig på MinaMeddelanden skickas meddelandet digitalt till mottagarens inkorg annars sker utskicket via papper och vanlig post.



2. Arkitekturens mål och restriktioner (Architectural Goals and Constraints)

I detta avsnitt beskrivs arkitekturens mål och de restriktioner som finns och därför måste tas hänsyn till vid utformandet av arkitekturen för systemet.

2.1 Restriktioner

Emrik ska begränsas till att endast stödja Adobe LiveCycle formatet för digitala utskick. Detta för att påskynda övergången från det äldre formatet Jetform.

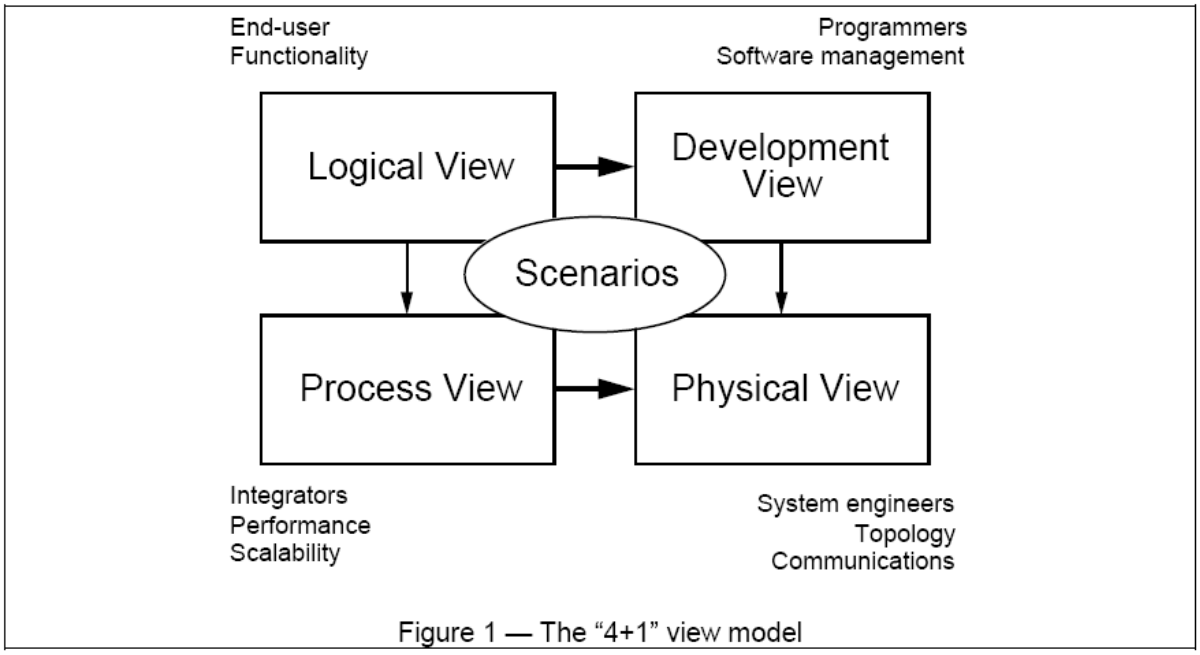
2.2 Målsättningar med arkitekturen

Id	*Mål*
M1	**Enkel och snabb anslutning** Arkitekturen skall bidra till att det är enkelt och snabbt att ansluta till MinaMeddelanden för Skatteverkets verksamhetssystem.
M2	**Stödja utveckling mot framtida målarkitektur** Arkitekturen skall styra mot uppfyllelse av framtida målarkitektur.
M3	**Skalbar lösning** Lösningen måsta vara byggd med mycket god skalbarhet i åtanke. Systemet skall klara av stora volymer med många transaktioner och många samtidiga användare i framtiden.

Tabell 2.1 Målsättningar med arkitekturen för ”Emrik”

3. Arkitekturens representation (Architectural Representation)

I detta avsnitt går vi igenom hur arkitekturen representeras och beskrivs i detta dokument. Beskrivningen av arkitekturen består av fem (4+1) olika vyer där varje vy är tänkt att fånga olika intressenters perspektiv på systemet. Modellen är tagen från *Architectural Blueprints—The "4+1" View Model of Software Architecture* [3] (figur 3.1).



Figur 3.1 Architectural Blueprints—The "4+1" View Model of Software Architecture

I detta dokument har vyerna översatts enligt tabell 3.2.

KUR namn	*RUP namn*	*Orginalnamn*
Logisk vy	Logical view	Logical view
Implementationsvy	Implementation view	Development view
Processvy	Process view	Process view
Driftsättningsvy	Deployment view	Physical view
Användningsfallsvy	Use-case view	Scenarios

Tabell 3.2 Översättning av vyer

3.1 Användningsfallsvy

3.1.1 Konceptuell nivå

På konceptuell och övergripande nivå i användningsfallsvyn tittar vi på systemet ur en slutanvändares perspektiv med fokus på systemets funktioner. Vi använder oss av enkla användningsfallsdiagram och andra modeller som beskriver dynamiken i systemet samt beskrivande text och skärmbilder. I detta dokument beskrivs inte alla användningsfall som systemet skall stödja, här fokuserar vi på användningsfall som är av speciell vikt för utformandet av arkitekturen och dess förmåga att uppfylla funktionella och kvalitetskrav.

3.1.2 Logisk nivå

På lägre och logisk nivå beskriver vi viktiga och fundamentala principer för systemets dynamik i olika scenarier vilka kan beskrivas i olika typer av sekvens- och tillståndsdiagram. På denna nivå beskrivs systemet främst ur designers och konstruktörers perspektiv.

3.2 Logisk vy

Den logiska vyn beskriver på olika nivåer systemet ur en slutanvändares och en systemutvecklares perspektiv. Här beskrivs informationen som hanteras av systemet och hur systemet är indelat i delsystem samt hur dessa delsystem kommunicerar med varandra och viktiga principer kring hur delsystemen fungerar. Dessa beskrivningar görs i vanliga klassdiagram och andra liknande modeller.

3.3 Implementationsvy

I implementationsvyn beskrivs systemet ur en konstruktörs och CM:s perspektiv. Här beskrivs hur systemets delsystem byggs upp av komponenter och strukturen för hur komponenterna byggs från källkod samt hur systemet kan delas upp i olika skikt. I denna vy beskrivs också andra systemexterna tjänster, komponenter eller system som skall användas vid realisering av systemet. Modellerna i vyn består av klass-, komponent- och skiktdiagram.

3.4 Processvy

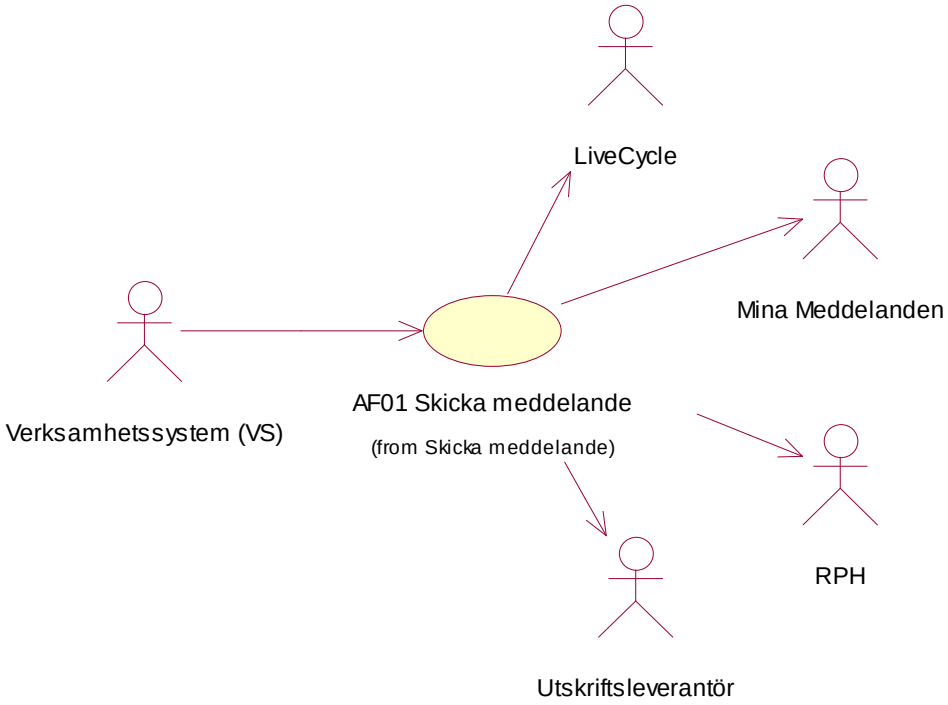
I processvyn beskrivs hur systemet hur systemet är uppdelat i processer och hur dessa processer kan delas upp på olika noder i ett nätverk. Om en process är uppdelad i trådar beskrivs även denna uppdelning här. Vyn beskriver hur olika exekverbara enheter (processer och trådar) av systemet arbetar parallellt och eventuellt behov av synkronisering dem emellan. Processvyn beskriver systemet ur perspektivet från konstruktörer, systemintegratörer och systemtekniker. I denna vy används klass-, sekvens- och tillståndsdiagram som beskrivande modeller.

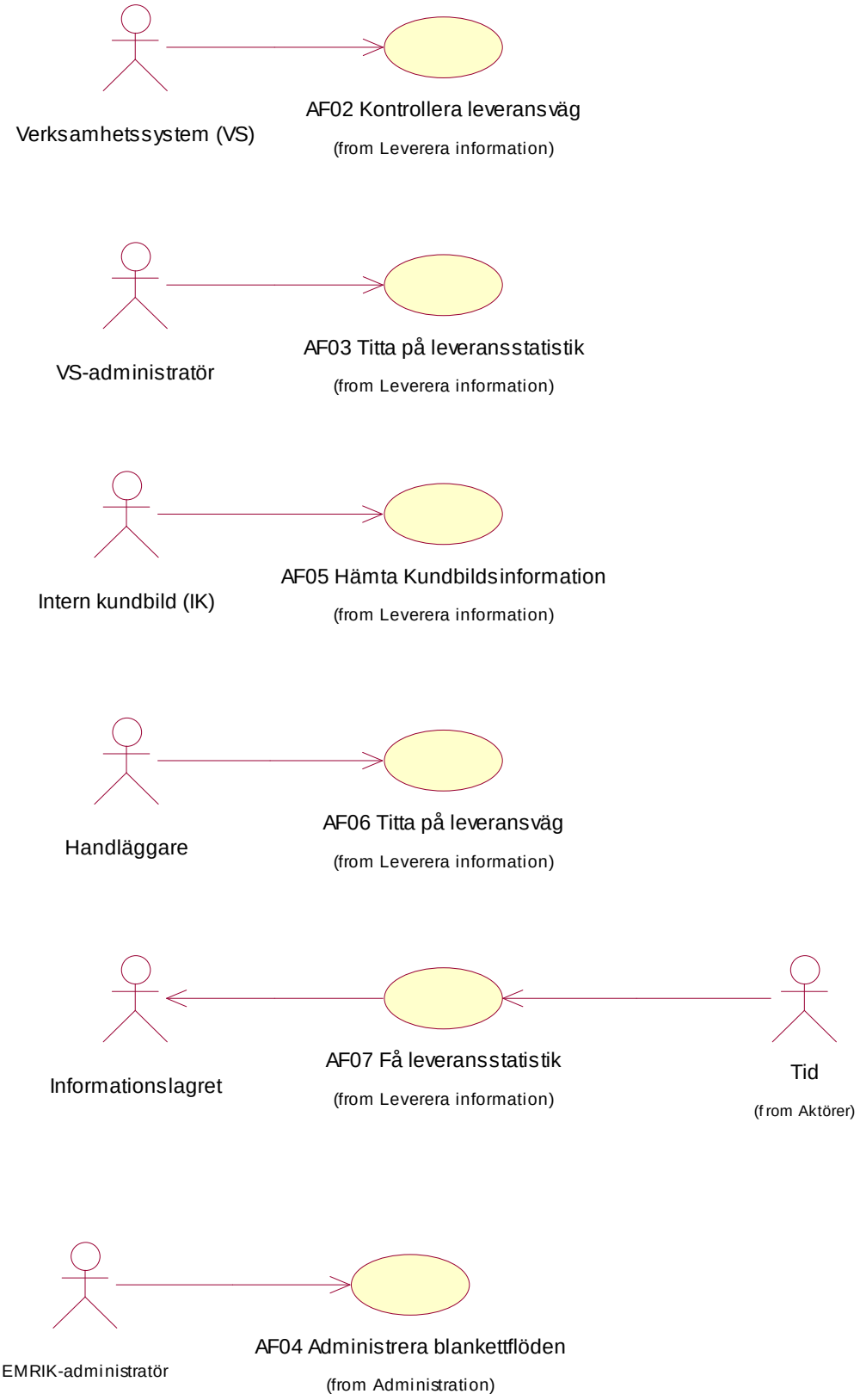
3.5 Driftsättningsvy

Driftsättningsvy beskriver infrastrukturen som krävs runt systemet för att systemet skall fungera i drift. Här beskrivs hur systemet installeras och driftsätts samt hur systemet är distribuerat och beroenden till externa system och komponenter. Protokoll som används för kommunikation mellan systemet och externa system samt säkerhetsmässiga aspekter beskrivs också här. Vyn beskrivs ur perspektivet från i första hand till systemtekniker och testare men vyn är intressant även för konstruktörer och arkitekter. Modeller som används är oftast deploymentdiagram.

4. Användningsfallsvy (Use-Case View)

I den här vyn redovisas de användningsfall i systemet som har bedöms som kritiska, dvs. de användningsfall som har störst påverkan på utformningen av arkitekturen. Utvalda användningsfall har används som hjälp för att driva fram detta arkitekturförslag.




Verksamhetssystem (VS)
AF02 Kontrollera leveransväg
(from Leverera information)
VS-administratör
AF03 Titta på leveransstatistik
(from Leverera information)
Intern kundbild (IK)
AF05 Hämta Kundbildsinformation
(from Leverera information)
Handläggare
AF06 Titta på leveransväg
(from Leverera information)
Informationslagret
AF07 Få leveransstatistik
(from Leverera information)
Tid
(from Aktörer)
EMRIK-administratör
AF04 Administrera blankettflöden
(from Administration)

Figur 4.1 Användningsfallsöversikt för Emrik

4.1 AF01 Skicka Meddelande

Användningsfallet används av Skatteverkets verksamhetssystem (VS) för att leverera meddelanden till mottagare. Om förutsättningar medger detta skapar systemet ett säkert digitalt meddelande och levererar detta till mottagarens konto på Mina meddelanden. Om förutsättningarna inte medger digitalt utskick skickas meddelandet för utskrift på papper

4.2 AF02 Kontrollera leveransväg

Användningsfallet beskriver hur VS använder systemet för att kontrollera leveransväg för enskilda skickade meddelanden eller meddelandepaket.

4.3 AF03 Titta på leveransstatistik

Användningsfallet beskriver hur en VS-administratör använder systemet för att titta på leveransstatistik för ett visst VS under en viss tidsperiod.

4.4 AF04 Administrera blankettflöden

Användningsfallet beskriver hur en EMRIK-administratör använder systemet för att:

- Administrera huruvida specifika blanketter ska kunna skickas digitalt till Mina Meddelanden eller om de alltid ska skickas till utskriftsleverantör.
- Lägga till blanketter
- Ta bort blanketter
- Ändra information om blanketter

4.5 AF05 Hämta kundbildsinformation

Användningsfallet beskriver hur systemet ”intern kundbild” använder systemet för att hämta information om alla meddelanden som har skickats till en enskild mottagare.

4.6 AF06 Titta på leveransväg

Användningsfallet beskriver hur en Handläggare använder systemet för att titta på leveransväg för enskilda skickade meddelanden eller meddelandepaket.

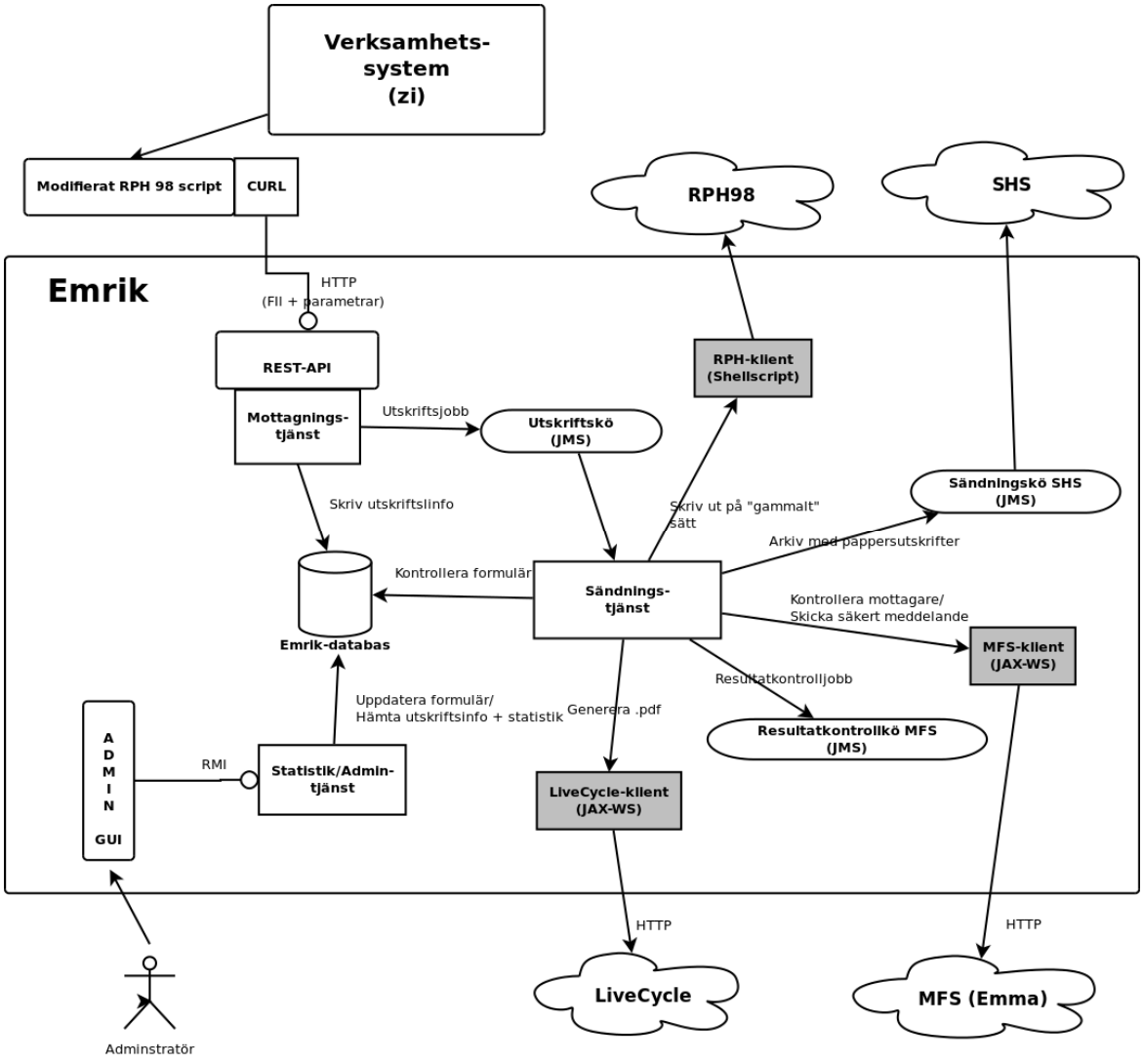
4.7 AF07 Få leveransstatistik

Användningsfallet beskriver hur systemet levererar information om alla meddelanden eller meddelandepaket som hanterats under en viss tidsperiod till informationslagret.

5. Logisk vy (Logical View)

5.1 Systemet dess intressenter och omgivande miljö

I figur 5.1 kan du se en översikt av ”Emrik” i sin omgivning med intressenter och aktörer samt de externa systemen som systemet är beroende av.



Figur 5.1 Översikt av Emrik och dess omgivning

5.1.1 Beskrivning av huvudsakligt flöde, enkel utskrift

När ett ZI-baserat verksamhetssystem vill göra en utskrift använder sig ZI:s utskriftskomponent av ett RPH98-shellscript. För att koppla in Emrik i flödet har ett modifierat script installerats på de system som vill nyttja Emrik. Det modifierade scriptet använder CURL för att anropa en REST-tjänst i Emrik som tar emot utskriften. Anropet innehåller en fil som ska skrivas ut och ett antal utskriftsparametrar.

Mottagningstjänsten gör följande när den får in en ny utskrift:

1. Kontrollerar om ett korrelations-id finns angivet i anropet, annars skapas ett nytt för retur till verksamhetssystemet. Korrelations-id:t kan senare användas för att göra uppföljning på utskriften.
2. Skriver den information om utskriften i Emriks spårningslogg, inkl en fail-safe kopia av filen för ev senare användning (se nedan)
3. Lägger upp ett utskriftsjobb på en intern utskriftskö.
4. Returnerar anropet.

En sändningstjänst betar asynkront av utskrifterna på utskriftskön. I kön kan det ligga utskrifter av 3 typer:

1. Jetform-utskrift (I form av ett XML-dokument LC-XML)
2. LiveCycle-utskrift (I form av ett XML-dokument JF-XML
3. Zip-arkiv innehållade flera LiveCycle-utskrifter

Utskrifter i Jetform-format kan i nuläget inte skickas för elektronisk utskrift. Dessa passas bara vidare till RPH genom ett anrop av en RPH-klient. Parametrar och fil som Emrik fick in passas vidare ograverat.

För utskrifter i Livecycle-format behandlas på följande sätt:

1. Kontrollera mot Emrik DB att formuläret som används är Ok för elektronisk utskrift, om ej Ok gör samma åtgärd som för Jetform (se ovan)
2. Kontrollera om verksamhetssystemet har angett att utskriften har ”Force paper”, d v s ska skrivas ut på papper oavsett. Om så är fallet gör samma åtgärd som för Jetform (se ovan).

Om båda kontrollerna ovan har passerats är utskriften en kandidat för att skickas som elektroniskt meddelande. Då extraheras mottagarens id (personnumret) från utskriftsfilen (Speciellt MFS-block) och ett anrop kan ske till MFS för att kontrollera om mottagaren har registrerat sig för att ta emot meddelanden elektroniskt. Anropet resulterar i ett svar som anger om det är Ok eller inte att skicka till mottagaren + om det inte är Ok en anledning till detta (Har ej konto, pågående registrering).

När MFS har svarat att det är Ok börjar själva sändningen av meddelandet. Först skapas ett .pdf-dokument genom att anropa LiveCycle och skicka med utskriftsfilen som inkluderar vilket formulär LiveCycle ska använda för att generera .pdf:en.

Den resulterade .pdf:en läggs i ett XML-dokument som representerar ett säkert meddelande som MFS kan ta emot. Dokumentet signeras med ett certifikat och skickas sedan till MFS.

Svaret som skickas när MFS mottagit meddelandet innehåller ingen status om hur skickadet gick. Status måste kontrolleras asynkront. Därför finns en resultatkontroll kö där ett jobb läggs upp för varje skickat meddelande som regelbundet (var 5:e sekund) används för att kontrollera status.

När status kommer in från MFS uppdateras informationen om utskriften i Emriks databas till att motsvara den nya statusen. Om skickandet misslyckas används den tidigare skapade ”Fail-safe”-kopian av utskriftsfilen för att göra en pappersutskrift via RPH.

5.1.2 Specialfallet arkiv med multipla LiveCycle-utskrifter

När ett Zip-arkiv innehållande flera LiveCycle-utskrifter betas av från utskriftskön packas arkivet upp och en lista med samtliga mottagar-id:n i arkivet extraheras. Listan skickas in i chunkar om 1000st till MFS som svarar med listor med mottagarstatusar som kan användas för att avgöra vilka filer som ska skickas digitalt resp. skrivas ut på papper. De som ska skickas digitalt läggs tillbaka på utskriftskön som enkeljobb och behandlas som beskrivet ovan. Pappersutskrifterna läggs tillbaka i arkivet och skickas via SHS för utskrift hos utskriftsleverantör (idag Parajett).

5.1.3 Specialfallet lokala utskrifter

Om en utskrift anges att skrivas ut lokalt skickas den direkt till RPH98. RPH98 har anpassats för att även kunna ta emot lokala utskrifter i LiveCycle-format.

5.1.4 Specialfallet förhandsgranskning Jetform

När det kommer in en Jetform-utskrift med korrelations-id satt till 1 är det en signal till Emrik som talar om att detta är ingen vanlig utskrift utan en begäran om en förhandsgranskning av Jetform-utskrift. En sådan passas bara vidare till RPH som hanterar förhandsgranskningen.

5.1.5 Admin-GUI

Emrik har ett administrations GUI som kan användas via en vanlig webblläsare. Via GUI:t kan Emriks systemadministratörer och personal från resp verksamhetssystem utföra diverse administrativa uppgifter. Via GUI:t kan man:

1. Administrera formulär som hanteras av Emrik : Lägga upp nya formulär och aktivera/inaktivera befintliga (se ovan hur inaktiva formulär hanteras).
2. Generera statistikfil: skapar en .csv-fil som innehåller en sammanställning på de utskrifter som har hanterats av Emrik filtrerat på verksamhetssystem och datumintervall.
3. Sök utskriftsjobb: med hjälp av korrelations-id:t kan man söka rätt på en utskrift och få en fullständig sammanställning av informationen som Emrik har om den utskriften.

5.2 Aktörer och intressenter

5.2.1 Verksamhetssystem

Ett system hos skatteverket som gör utskick och som önskar göra dessa elektroniskt där möjligheten finns. Använder Emrik för att ansluta sig till MFS.

5.2.2 Administratör

Antingen en systemadministratör i Emrik eller personal som jobbar med något av verksamhetssystemen och önskar ta del av Emriks information om utskrifter.

5.3 Externa beroenden

5.3.1 LiveCycle

Produkt från Adobe som bl a kan generera .pdf-dokument baserat på en utskriftsfil och ett i förhand designat formulär.

5.3.2 MFS

MFS(Meddelandeförmedlingssystemet) är ett system som tillåter personer och företag att registreras för att ta emot meddelanden från myndigheter elektroniskt och sedan ser till att meddelanden levereras till personen/företagets elektroniska brevlåda. Det enda MFS som finns idag är det som implementerats av Skatteverkets projekt ”Emma”, men i framtiden kommer andra aktörer att kunna agera MFS.

5.3.3 RPH

Äldre utskriftssystem hos Skatteverket. Hanterar i dag både lokala och centrala utskrifter, men kommer i framtiden att endast hantera lokala. En Zi-komponent för utskrifter används som ingång till Zi genom att tillhandahålla ett Java-API för anrop av de shellscript som annars är gränssnitt mot RPH.

5.3.4 SHS

SHS är en standard inom statlig förvaltning [6] för säkert informationsutbyte mellan olika organisationer. Det finns flera leverantörer som erbjuder produkter som implementerar standarden vilka kan avropas enligt ramavtal [7] för ”Infratjänst”.

SHS finns installerade på ett flertal myndigheter, länsstyrelser och kommuner.

5.4 Informationsmodell

Figuren 5.7 nedan illustrerar Emriks interna informationsmodell bestående av 3 entitetsobjekt. Informationsmodellen är rak, utan relationer mellan objekten.

5.4.1 DeliverySentToMfsEO

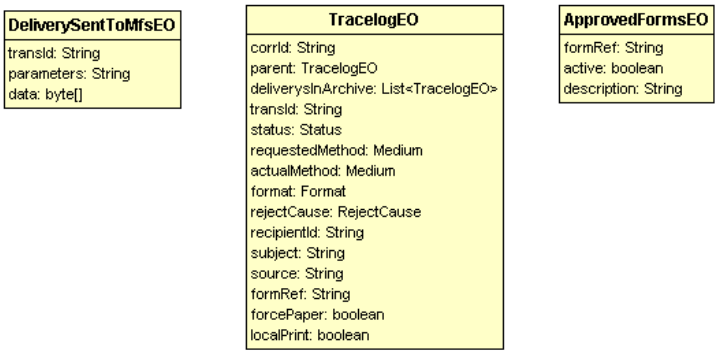
Används för att göra en ”Fail-safe”-kopia av filen och associerade parametrar för att kunna göra en pappersutskrift om den digitala skulle misslyckas.

5.4.2 TracelogEO

Emriks spårningslogg för utskriftsjobb. Kan användas för att följa upp utskrifter genom korrelations-id:t (corrId). Håller information som status på jobbet, utskriftsmetod(önskad och verklig), transaktions-id från MFS, mottagare, flaggor för tvingande pappersutskrift och lokal utskrift och en lista av innehållade jobb om utskriften är ett arkiv.

5.4.3 ApprovedFormsEO

Används för att lagra formulär som är godkända för elektronisk utskrift. Har en flagga som kan användas för att tillfällig slå av/på elektronisk utskrift för ett visst formulär.



Figur 4.1 Informationsmodell för Emrik

6. Implementationsvy (Implementation View)

6.1 Översikt (Overview)

Emrik kan delas in i skikt och ett skikt kan bestå av flera lager, se figur 6.1. I detta dokument använder vi följande definition av “skikt” och “lager”:

- **Lager** – logisk uppdelning av funktionalitet, d v s en gruppering av moduler med avseende på beroenden, där ett lager bara är beroende av det underliggande.
- **Skikt** – uppdelning av funktionalitet på sådant sätt att det är möjligt för de olika skikten att köras på olika datorer och olika plattformar.

Systemet delas också in i delsystem och denna indelning är gjord för att hålla samman ”logiska enheter av funktionalitet” för systemet. I Emrik skall ett delsystem hållas inom ett och samma skikt[1] så att man fritt kan sprida delsystemen på olika maskiner och nätverksnoder. Längre fram i detta avsnitt beskrivs hur indelningen är gjord för Emma. Delsystemen byggs upp av komponenter som implementeras av klasser, konfiguration och andra resurser.

[1] Delsystem i presentationslogikskiktet kommer att omfatta kod som körs i klientskiktet. Eftersom denna kod ”laddas ner” till klienten och därför inte kräver någon installation bryts inte principen om distribution på olika noder.

6.2 Skikt och lager (Tiers and layers)

I figur 6.2 illustreras skikt- och lagerindelningen för Emma. I denna vy beskriver vi ansvaret i varje skikt/lager och vilken teknik som används för att implementera logiken samt hur delsystemen fördelar sig.

6.2.1 Klient

I klientskiktet finns de verksamhetssystem som anropar Emrik via REST-gränssnittet for att göra utskrifter. I klientskiktet placerar vi även in Admin-GUI:t där användare kan via en webbläsare administrera systemet. .

6.2.2 Integration

I integrationsskiktet finns SHS som kan användas vid kommunikation mellan myndigheter och andra organisationer.

6.2.3 Presentationslogik

Presentationslogikskiktet ansvarar för att rendera och leverera resurser till klienten (webläsaren) via http-protokollet. Resurserna kan vara av olika typer, men de vanligaste är HTML och XML. Presentationslogiken implementeras med JSP och Strutsramverket [11] samt AJAX-teknik. JSP och Struts implementationen skapar HTML som används för att rendera gränssnittet på klienten. I detta skikt sker även transformering av information från systemets tjänster till XML som används tillsammans med AJAX-teknik för att uppnå funktionalitet i klienten som mer liknar en klassisk ”fet klient”.

6.2.4 Verksamhetslogik

I verksamhetslogikskiktet finns all verksamhetslogik som exponeras via systemets tjänster.

6.2.5 Tjänster

Tjänstelagret ansvarar för att exponera tjänster genom REST.

Tjänsterna implementeras som EJB-3 stateless session beans vilka exponeras med *@webservice* annotering. Genom att implementeras tjänsterna som EJB:er erhålles stöd för persistenscontext och deklarativa transaktioner automatiskt.

Tjänsterna delas in i grupperna *publika* och *privata*. Publika tjänster är tjänster som görs tillgängliga för anslutande organisationer. Dessa tjänster skall uppfylla de tekniska tjänstekontrakten [21] som gemensamt specificerats av de samarbetande myndigheterna (SKV, BOL, TVV) inom verksamt.se programmet. Privata tjänster är tjänster som används internt av Emrik-systemet.

6.3 Paketering av delsystem och komponenter

Ändra till motsv Emrik

I figuren nedan beskrivs hur systemet paketeras i olika arkiv. Delsystemen ”Rest” och ”Admin”) paketeras som en webapplikation i ett ”Web application archive” (WAR) [12]. Webarkiven omfattar egna UI (User Interface) komponenter samt komponenter från tredje part. Webarkiven inkluderar även de komponenter som delas över skiktsgränsen. Verksamhetslogikskiktets delsystem (backend) paketeras i ”Enterprise application archive” (EAR) [9]. Dessa arkiv innehåller tjänsterna och persistenshanteringen samt delade komponenter. Övriga komponenter paketeras i vanliga Java arkiv (JAR) [13].

Det är viktigt att varje delsystem paketeras enligt beskrivningen ovan eftersom man vill uppnå att varje delsystem kan lösa upp sina beroenden på klassladdaren som laddar delsystemet. Beroenden till klasser som ingår i JRE är de enda klasserna som får laddas av systemklassladdaren. Genom denna paketering kan nya versioner av delsystemen driftsättas utan att systemet måste startas om. Risken för versionskonflikter map olika versioner av samma klass minimeras också.

6.4 Katalogstruktur för källkod

I figurerna 6.10a-c nedan illustreras en översikt över katalogerna som innehåller delsystem, komponenter och klasser som bygger upp Emrik. Den totala katalogstrukturen innehåller fler kataloger som innehåller byggscript, modeller, dokumentation osv. Katalogstrukturen bygger på översta nivån på den standard som används på Skatteverket för alla Javasystem. I nivåerna under den översta nivån följer katalogstrukturen Maven-standard.

6.4.1 models

Innehåller alla tjänstekontrakt och schema specifikationer.

6.4.2 libs

Innehåller alla komponenter som delas mellan presentations- och verksamhetslogik-skikten.

6.4.3 subsystems

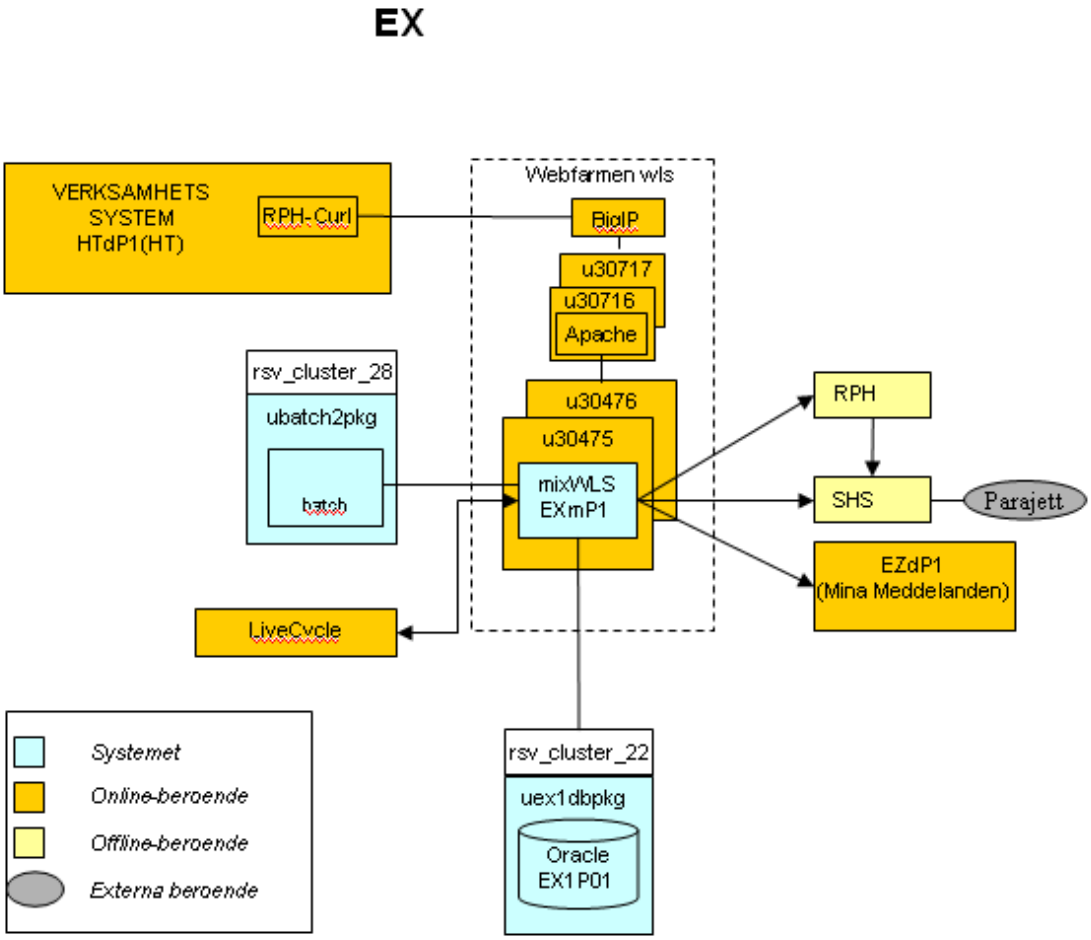
Innehåller alla delsystem som tillhör verksamhetslogikskiktet.

6.4.4 web

Innehåller alla delsystem som tillhör presentationsskiktet.

7. Driftsättningsvy (Deployment View)

Bilden nedan illustrerar hur driftsmiljön för Emrik ser ut. Det första verkamhetssystemet Hunten är inritat som exempel. Alla delsystems WLS-domäner (*P1) och ingående servrar (u*) är angivna



7.1 Valda programvaror

Tabellen nedan visar valda programvaror för Emrik.

Namn	Version	Beskrivning
Oracle Weblogic Server	10.3	Applikationsserver med fullt stöd för Java Enterprise Edition 1.5
Oracle RDBMS	11g	Relationsdatabas
Oracle JRE	1.6	Java runtime
HPUX	11.2	Operativsystem
iipax	4.4	SHS - Spridning och hämtningssystem
Apache	2.2	Webbserver

Tabell 8.1 Valda programvaror

8. Datamängd och prestanda (Size and Performance)

Här redovisas kraven som ställs på systemet avseende datamängd och prestanda.

8.1 Nuvarande kortare toppar i last för meddelanden

Systemet ska klara av att hantera och leverera 5 000 meddelanden per timme under 2 timmar under följande förutsättningar:

- 22 % av meddelanden består av 13 kB stora medelanden till utförare. Av dessa ska c:a 10 % gå till utförare med konto i Mina Meddelanden.
- Övriga 78 % ska bestå av c:a 4 kB stora meddelanden till köpare, varav c:a 1 % ska gå till köpare med konto i Mina Meddelanden.
- Alla meddelanden uppfyller kraven för att kunna användas till att generera meddelanden (formatet är ALC, blanketten är tillåten för digitala meddelanden och meddelandena har inte ForcePaper=TRUE).

8.2 Nuvarande längre toppar i last för meddelanden

Systemet ska klara av att hantera och leverera 40 000 meddelanden på 10 timmar .

8.3 Nuvarande dygnslast för meddelanden

Systemet ska klara av att hantera och leverera 60 000 meddelanden på 24 timmar .

8.4 Nuvarande lokala utskrifter

Systemet ska klara av att hantera och leverera 300 lokala utskrifter per timme.

8.5 Nuvarande svarstider för ”kontrollera leveransväg”

Systemet ska klara av att hantera 10 000 förfrågningar om leveransväg per timme.

8.6 Framtida kortare toppar i last för meddelanden

Systemet ska i framtiden kunna klara av att hantera och leverera 7 000 meddelanden per timme under 8 timmar under följande förutsättningar:

- 22 % av meddelandena består av 13 kB stora medelanden till utförare. Av dessa ska c:a 30 % gå till utförare med konto i Mina Meddelanden.
- Övriga 78 % ska bestå av c:a 4 kB stora meddelanden till köpare, varav c:a 5 % ska gå till köpare med konto i Mina Meddelanden.
- Alla meddelanden uppfyller kraven för att kunna användas till att generera meddelanden (formatet är ALC, blanketten är tillåten för digitala meddelanden och meddelandena har inte ForcePaper=TRUE).

8.7 Framtida dygnslast för meddelanden

Systemet ska klara av att hantera och leverera 70 000 meddelanden på 24 timmar under samma förutsättningar som i kapitel 8.6.

8.8 Framtida svarstider för ”kontrollera leveransväg”

Systemet ska klara av att hantera 14 000 förfrågningar om leveransväg per timme.

9. Kvalitet (Quality)

I detta avsnitt går vi igenom alla kvalitetsegenskaper och tittar på vilka egenskaper som är speciellt viktiga för Emma's samt hur arkitekturen bidrar till att rätt kvalitet kan uppnås. Vi kopplar även tillbaka till målsättningarna för arkitekturen och tittar på uppfyllanden av dessa mål.

9.1 Kvalitetsegenskaper

Förklaring av vad de olika kvalitetsegenskaperna innebär.

Kvalitetsegenskap	Översatt	Beskrivning
Usability	Användbarhet	*Hur enkelt är det för användaren att förstå och använda systemet*
Security	Säkerhet	*Systemets förmåga att hindra obehöriga försök att komma åt systemets data och DOS-attacker samtidigt som systemet erbjuder service till behöriga användare*
Availability	Tillgänglighet	*Tiden som systemet är uppe och fungerar. Mäts som tiden mellan fel och hur snabbt systemet kan startas om efter fel*
Performance	Prestanda	*Ett mått på systemets svarstid för en funktion*
Modifiability	Förändringsbarhet	*Ett mått på hur enkelt det är att förändra systemet för att omfatta ny funktionalitet. Två aspekter på förändringsbarhet är tid och kostnad. Om ett system är dyrt att ändra (specialistkunskaper) men det går fort så är ändå förändringsbarheten låg.*
Testability	Testbarhet	*Hur enkelt är det att testa systemet, manuellt eller maskinellt. God testbarhet är relaterat till hur modulärt systemet är. Om systemet består av komponenter med väldefinierade gränssnitt borde testbarheten vara god.*

Tabell 6.1 Kvalitetsegenskaper (källa Enterprise Architecture, J. McGovern mfl.)

Nedan följer en lista med en beskrivning av kvalitetsegenskaper och deras prioritering för Emrik-systemet. Prioriteringen har gjorts i samråd mellan representanter för respektive myndighet i samarbetet kring Mina meddelanden och skall användas som vägledning när frågor kring arkitekturval skall beslutas.

1. Systemet har mycket höga skalbarhetskrav
   Begreppet prestanda används här i betydelsen att många kan använda systemet samtidigt och att den totala mängden meddelanden i systemet kan bli mycket stort.

2. Integrationsmöjligheter
   Systemet består av samverkande tjänster från olika myndigheter, ska kunna knytas till olika mötesplatser/webbplatser och ställer därför höga krav på enkla och tydliga sätt att utbyta information mellan de betjänande parterna.

3. Höga krav på säkerhet (Säkerhet)
   Majoriteten av användare har verifierad identitet vars meddelanden måste skyddas.

4. Mycket höga tillgänglighetskrav
   Begreppet tillgänglighet används här i betydelsen att användaren kan nå sina meddelanden vid de tidpunkter som passar de egna förutsättningarna utan begränsningar av Expedierande myndighet eller andra.

5. Robust applikation
   Systemet har mycket höga krav när det gäller användarnas tillit till systemet och dess stabilitet. Tekniska driftstörningar i delar av tekniska miljön ska inte stoppa användarens hela session, tydliga fel- och konsekvensbeskrivningar ska alltid ges.

6. Förändringsbarhet
   Systemet består av samverkande tjänster från olika myndigheter och ställer därför höga krav på förändringsbarhet.

7. Återanvändbarhet
   Systemet ska tillföra stor nytta per investerad krona och i sin informationsarkitektur utformas så att dessa delar kan leva under mycket lång tid. Presentationsdelar och transportmekanismer i systemet bedöms ha kortare livslängd och kan därför prioriteras lägre.
   Andra viktiga systemegenskaper, till exempel enskilda svarstider, skiljer sig mindre från befintliga system och förväntas därför inte i samma omfattning medföra ökade risker i detta system. Av detta följer att vi prioriterar att säkerställa ovanstående icke-funktionella egenskaper framför andra då systemarkitekturen fastställs och att testfall gentemot dessa krav får högre prioritet än andra.

9.2 Motivering av arkitekturens mål

I detta avsnitt beskriver vi hur väl arkitekturen som beskrivits i föregående avsnitt uppfyller målen som beskrivits i avsnitt *2.2 Målsättningar med arkitekturen.*

I tabell 6.2 går vi igenom alla mål och bedömer hur väl målet uppfylls enligt följande skala: *0 – Inte alls, 1 – Delvis, 2 –Mycket väl*
Bedömningen av varje mål motiveras i kolumnen ”Kommentar” med en kort beskrivande text samt en referens till var detta beskrivs i arkitekturbeskrivningen.

Id	*Mål*	*Prio*	*Kommentar*	*Uppf.*
M1	**Enkel och snabb anslutning**	1	Använder befintlig ZI-utskriftskomponent. Den enda ändring som krävs är att VS konverterar sina formulär till LC_XML	2
M2	**Stödja utveckling mot framtida målarkitektur**	1	Stödjer övergång från Jetform till LiveCycle genom att tvinga VS att konvertera om digitala utskick ska kunna göras.	2
M3	**Skalbar lösning**	1	Bygger på JEE-teknik med helt separerade delsystem vilket ger en effektiv möjlighet till skalning av systemet.	2

Tabell 6.2 Motivering av måluppfyllnad för arkitekturen

9.3 Revisionshistorik

Här redovisas dokumentets revisionshistorik.

Datum	**Version**	**Beskrivning**	**Signatur**
2013-01-16	0.1	Skapat dokumentets grundstruktur	RP
2013-01-28	0.2	Uppdaterat efter synpunkter från Toni Thomsson	RP

Tabell 7.2 Revisionshistorik för dokumentet

9.4 Dokumentansvarig

Ansvarig för detta dokument och dess innehåll är projektet ”*Emrik* (EC)”.
Synpunkter eller frågor med anledning av dokumentet lämnas till *Ronnie Pettersson.*

Dokumentets filnamn är *ec_emrik_sad.doc*