
Fakulteten för ekonomi, kommunikation och IT

Johan Björlin

# En ansats till behovsstyrd applikationsutveckling

## An approach to needs-based application development

Datavetenskap
C-uppsats (15 hp)

Datum/Termin: 2011-06-09
Handledare: Kerstin Andersson
Examinator: Donald F. Ross
Löpnummer: C2011:05

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se


Datavetenskap

---

**Johan Björlin**

# En ansats till behovsstyrd applikationsutveckling

---

Examensarbete, C-nivå

CPubnum, Arkiv | Egenskaper | Eget

# En ansats till behovsstyrd applikationsutveckling

**Johan Björlin**

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Johan Björlin

Godkänd, 2011-06-09

Handledare: Kerstin Andersson

Examinator: Donald F. Ross

# Sammanfattning

Försvarsmaktens Telenät och Marktele Förband, FMTM, behövde ersätta en gammal applikation för att hantera vissa typer av personal. Idén var att den nya applikationen skulle kunna säkerhetsgodkännas och möjliggöra att flera användare skulle kunna arbeta i den samtidigt, vilket inte var fallet med den tidigare. De data som fanns i applikationen skulle också kunna merutnyttjas, dels för Vakthavande befäl som behövde en möjlighet att söka efter data om en persons anhöriga och dels som en telefonkatalog. Utvecklingsprocessen följde ingen given modell men lånande inslag från Unified Process, UP och dokumenterades i Unified Modeling Language. Genom dialog med beställarens representant och analys av den befintliga applikationen skapad i Microsoft Access som inte är en av Försvarsmakten godkänd programvara skapades en kravbild. I denna kravbild fanns funktionella och icke-funktionella krav där de funktionella kraven förtydligades med en användningsfallsmodell och som sedan genomgick en analys för att få fram en arkitektur för applikationen som döptes till Personnel Management System, även kallad PMS. Genomförda avgränsningar innebar att bara applikationsdelen för personalhantering initialt skulle skapas. I analysen hanterades också de icke-funktionella kraven. Analysen resulterade i en lösning skriven i programspråket C# med en serverdel baserad på en webbtjänst, en WindowsFormsbaserad klient och data lagrat i en SQL-databas. Efter analysen skapades en design utifrån vilken en implementation med en begränsad funktionalitet genomfördes. Verifiering och validering har ännu inte genomförts. Ansatsen i utvecklingsarbetet har varit att försöka realisera användarnas behov snarare än att generera ”perfekt” kod. Men då processen tagit tid finns en farhåga att användarnas målbild med systemet ska ha förändrats på ett sådant sätt att även om kraven kan verifieras är det inte säkert att användarna känner att det är rätt system då de under resans gång själva blivit klokare på vad det är de behöver.

# An approach to needs-based application development


## Abstract

The Swedish Armed Forces Network and Telecommunications Unit, FMTM, needed to replace an old application to manage certain categories of staff. The idea was that the new application should be security approved and also allow multiple users to work at the same time, which was not the case with current application. Data contained in the application should be used for other purposes as well, for Duty Officers who need an opportunity to search for data about a person's relatives and as a telephone directory. The development process followed no set model but borrowed elements from the Unified Process, UP, and was documented using the Unified Modeling Language. Through dialogue with the employer's representative and analyzing the existing application created in Microsoft Access that is not authorized software in the Swedish Armed Forces, requirements of both functional and non-functional nature was created. The functional requirements were clarified using a Use-Case that went through an analysis to derive the architecture for the application that was named the Personnel Management System, also known as PMS. Demarcations made the work only to be about creating the application portion containing the personnel management part. The analysis also dealt with the non-functional requirements. The analysis resulted in a solution written in C # with a server based on a Web service, a Windows Forms-based client and data stored in a SQL database. Based on the analysis a design was created from which the application was implemented with limited functionality. Verification and Validation has not yet been done. The approach in the development work has primarily been to try to realize the needs of users rather than to generate "perfect" code. But as the process took time, there is a concern that users' goals with the system should have changed in such a way that even if the requirements can be verified, it is not certain that the users feel that” it is the right system” when the journey has changed the idea of what their needs are.

# Innehållsförteckning

# Figurförteckning

# Tabellförteckning

# 1 Inledning

Detta arbete avhandlar utvecklingen av en ersättningsapplikation för att hantera personal. Fokus i arbetet ligger på att fånga användarnas behov och idéer och via krav och modeller omvandla dessa till en applikation som ligger nära användarnas egen målbild.

## 1.1 Bakgrund

Personalavdelningen (J1) på Försvarsmaktens Telenät och Markteleförband, FMTM, ”ett ständigt insatt insatsförband, såväl nationellt som internationellt, i hela beredskapsskalan” [1] hanterar dagligen en stor mängd personal som inte inryms i Försvarsmaktens ordinarie personalhanteringssystem PRIO [2]. Denna personalgrupp utgörs i huvudsak av tillsyningsmän och frivilliganställda. Data om denna personal lagras i dag i en äldre Microsoft Access Databas som finns på en bärbar laptop eftersom Microsoft Access inte är en godkänd programvara i Försvarsmakten och därmed inte får köras i Försvarsmaktens nätverk.

## 1.2 Syfte

Syftet med detta arbete är att påbörja skapandet av en ersättning för ovan nämnda Microsoft Access Databas. Ersättningssystemet ska kunna hantera flera samtidiga användare från personalavdelningen. Data som är lagrade i databasen ska även kunna användas som en telefonkatalog. Denna telefonkatalog är tänkt att kunna användas av alla delar av FMTM:s organisation. Behov finns också för Vakthavande Befäl (VB) på FMTM att ur databasen söka ut den som är närmaste anhörig till en person, något som ska kunna användas för att lättare komma i kontakt med anhöriga vid till exempel en olycka. Ersättningssystemet kan således sägas bestå av tre olika delar som använder samma data, en personaladministrativ del, en anhörigdel och en telefonkatalog.

## 1.3 Målsättning

Målsättningen för ersättningssystemet är främst att bygga bort begränsningar som funnits i den tidigare applikationen i Microsoft Access, varav de viktigaste är att öka tillgängligheten och att möjliggöra lagring av mer data om respektive person som hanteras av systemet.

## 1.4 Metod

Då utvecklingsarbetet endast genomförs av en person kommer ingen specifik utvecklingsmodell att användas under utvecklingen. De olika ingående processdelarna som ändå genomförs under arbetet kommer därför att beskrivas för varje delprocess från kravinsamling, via analys och design till implementation. Utvecklingen kommer att genomföras iterativt och inkrementellt, något som överensstämmer med många av de utvecklingsmodeller som praktiseras idag. Detta stämmer också överens med tankar om så kallade ofullständigt formulerade problem och de teorier som finns om hur dessa kan lösas [3]. Utvecklingen kommer medföra att allt eftersom produkten färdigställs och kundens förståelse för sina behov förtydligas så kan kraven på systemet förändras och nya idéer arbetas in, detta är dock en process i vilken detta arbete bara utgör en inledning.

Resterande del av detta dokument avser beskriva de olika ingående delarna av den använda utvecklingsprocessen och resultatet av dessa. I kapitel 0 avhandlas kraven som ställs på applikationen, i kapitel 0 beskrivs den användningsfallsmodell som används för att realisera kraven i föregående kapitel. I kapitel 0 analyseras hur applikationen ska kunna realiseras utifrån krav, användningsfallsmodell och användarinteraktion. Genomförandet beskrivs sedan i kapitel 0. Kapitel 0 och kapitel 0 kan sägas beskriva vad som ska göras, medan kapitel 0 och kapitel 0 övergår till att beskriva hur det ska göras och själva genomförandet. I kapitel 0 återges avslutningsvis de slutsatser som dragits under och efter utvecklingsarbetets genomförande.

## 1.5 Avgränsningar

Det står tidigt under arbetet klart att framtagandet och leveransen av en fullständig produkt inte kommer kunna genomföras under tiden för detta arbete. Kundens representant har därför prioriterat vilka funktioner som är viktigast inledningsvis och vad som kan vänta till senare. Således handlar detta arbete snarare om att leverera en prototyp som utgör en grund för ett fortsatt utvecklingsarbete.

Prioriteringen innebär att leveransen delas upp i olika steg med gradvis ökande funktionalitet.
I steg 1 prioriteras grundläggande funktionalitet i den personaladministrativa delen av applikationen, det vill säga hanterandet av data om en person och dennes adresser, telefonnummer, e-postadresser, ID-kort, pass, registerkontroller, militära förarbevis, körkort,

organisatorisk tillhörighet och anhöriga. I steg 2 skiftas fokus mot ett färdigställande av anhörigdelen av applikationen. Sedan i steg 3 tillförs funktionalitet för att hantera utbildningar, militär verksamhet, andra eventuella anställningar tillsammans med telefonkatalogsdelen av applikationen.

Fortsättningsvis kommer i detta dokument i huvudsak bara implementation av steg 1 att avhandlas då implementation av steg 2 och steg 3 inte kommer att vara genomförd vid tidpunkten för detta arbetes färdigställande.

Vad gäller förhållningssättet till huruvida det som ska lagras om en person i applikationen rör sig om data eller information så har författaren valt att applicera Peter Checkland och Sue Holwells [4] perspektiv, de menar att det som hanteras utgörs av data och att informationen skapas av användaren på det kognitiva planet snarare än att data blir till information genom de relationer som beskrivs i databasen. Följaktligen kommer därför de uppgifter som hanteras av applikationen om till exempel en person relateras till som data.

# 2 Krav på systemet

För att komma fram till vilka krav verksamheten ställer på den nya applikationen fördes en löpande dialog med en av beställaren utsedd representant. Processen för kravframtagandet kan sägas till stora delar följa hur Sven Eklund och Hans Fermlund i boken ”Programkonstruktion med kvalitet – projekthantering och ISO 9000” från 1998 [5] menar att kravframtagandet bör genomföras. De menar att krav framtagna genom dialog minskar risken för missförstånd och beskriver även ett antal punkter som leverantören bör ta hänsyn till. Av dessa punkter kan intervjuer med användare, analys av existerande system, studier av omgivningen för att identifiera speciella krav, nivåindelning av krav, framtagande av enkelt förståelig dokumentation och kundens delaktighet i kraven sägas ingå i detta arbete.

Dialogen med beställarens representant avhandlade således det befintliga systemet, hur det fungerade, vad som upplevdes bra och vad som upplevdes dåligt och vilka övriga önskemål som fanns på ett nytt system. Dialogen rörde också vilken data som det fanns behov av att lagra om de personer som systemet skulle användas för att hantera och hur processen för att inhämta dessa data vanligast tar sig uttryck. I resten av detta kapitel beskrivs den uttolkade kravbilden för applikationen.

Kraven är skrivna med en ansats att de ska vara enkla att förstå, att varje krav bara ska beskriva endast en sak och att förståelsen av kravet inte ska vara beroende av något annat krav. Kravbilden kan upplevas som onödigt detaljerad men syftet är också att det ur ett mätbarhetsperspektiv ska gå att svara ja eller nej på frågan huruvida ett krav är uppnått eller inte. Detta för att undvika situationer där ett krav till exempel skulle kunna vara uppnått till 65,3 %, och då det inte kan anses att det är uppfyllt, något som kan inträffa om ett krav innefattar för mycket.

Kraven har i resterande del av detta kapitel delats upp i funktionella krav (2.1) som beskriver de funktioner som ska ingå i applikationen och icke-funktionella (2.2) krav som kan sägas vara ramfaktorer för applikationen.

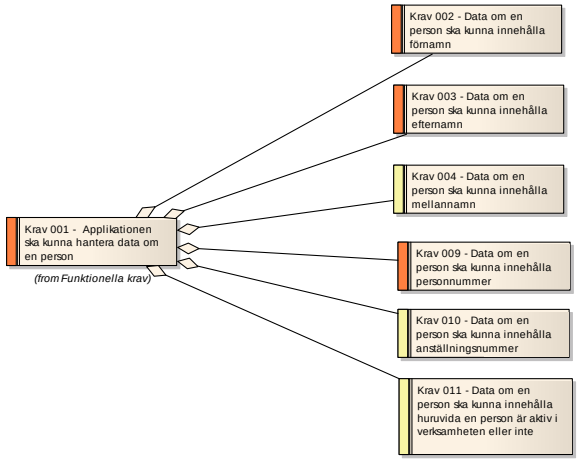
För att underlätta hanteringen av kraven används modelleringsverktyget Enterprise Architect tillverkat av Sparx Systems [6]. Verktyget har också använts för användningsfallsmodellen som beskrivs i kapitel 0.

## 2.1 Funktionella krav

Detta avsnitt beskriver de krav som ställs på applikationens funktioner. Enligt Wikipedia så definieras inom software engineering ett funktionellt krav som definitionen av en funktion för ett mjukvarusystem eller dess komponent där en funktion i sin tur beskriver ett antal input, ett beteende och outputs. Funktionella krav kan vara beräkningar, tekniska detaljer, datamanipulation och processning och annan specifik funktionalitet som definierar vad systemet förväntas åstadkomma [7]. Funktionella krav specificerar alltså specifika resultat för ett system. Kraven som beskrivs nedan handlar i huvudsak om datamanipulation och om vilken typ av data som applikationen är tänkt att hantera.

### 2.1.1 Krav på grundläggande personattribut

Kraven i detta avsnitt är avgränsade till att endast röra hantering av de personspecifika data som kan sägas vara det som delvis utgör en person snarare än något personen har. Dessa data som i fortsättningen benämns grundläggande personattribut utgör bara en delmängd av de data som används av de olika delarna av applikationen.




*Figur 1: Krav på grundläggande personattribut*

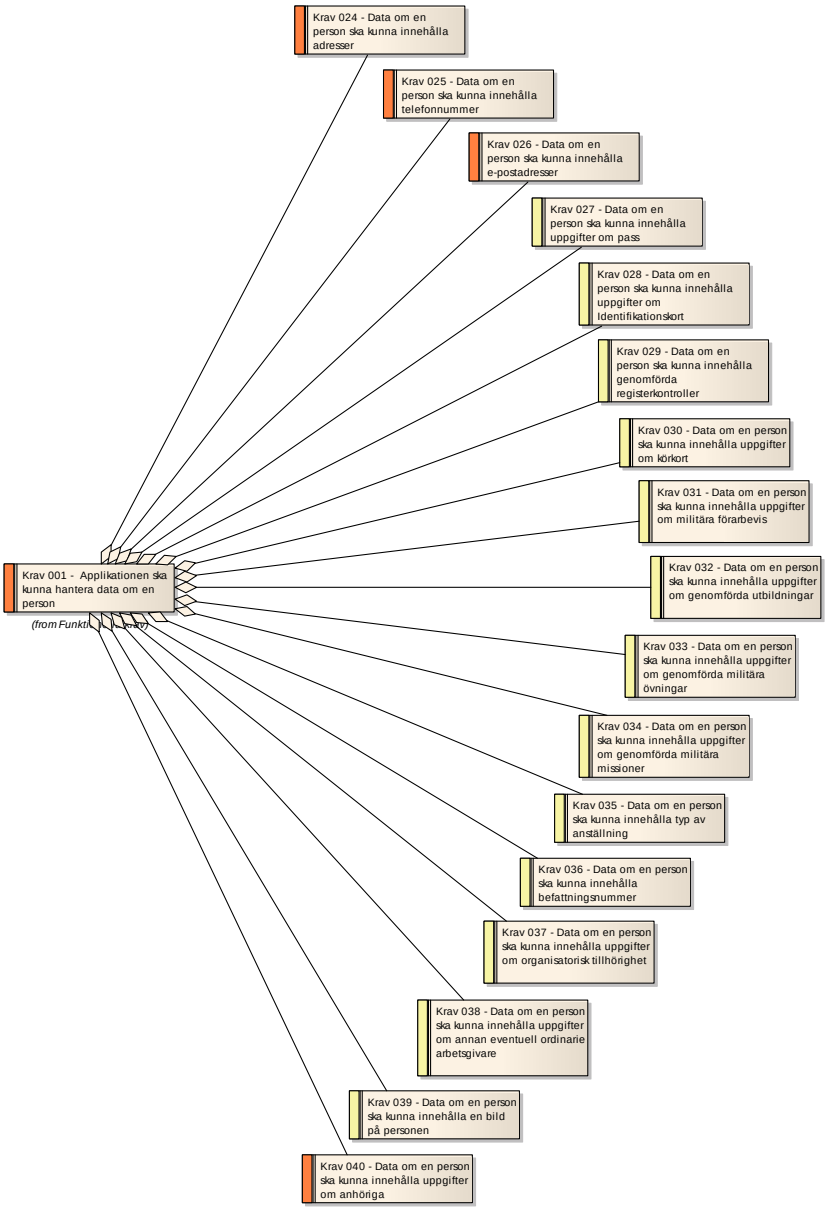
Hela syftet med applikationen är att kunna hantera data om personer, därför har ett övergripande krav för detta skapats. Detta krav, ”Krav 001 – Applikationen ska hunna hantera data om en person”, har sedan delats upp i krav som specificerar de data om en person som ska lagras. Med hjälp av verktyget för kravhantering byggs en kravhierarki, som delvis visas i Figur 1, vilket medför att för att kraven på de övergripande nivåerna ska anses uppfyllda så måste underliggande krav också vara det. För att Krav 001 ska vara uppfyllt så måste även ”Krav 002 – Data om en person ska kunna innehålla förnamn”, ”Krav 003 – Data om en

person ska kunna innehålla efternamn”, ”Krav 009 – Data om en person ska kunna innehålla personnummer” även vara uppfyllda. Andra krav som också måste vara uppfyllda för att Krav 001 ska anses vara uppnått är ”Krav 004 – Data om en person ska kunna innehålla mellannamn”, som är mindre viktigt än till exempel förnamn eller efternamn, ”Krav 010 – Data om en person ska kunna innehålla anställningsnummer” och ” Krav 011 - Data om en person ska kunna innehålla huruvida en person är aktiv i verksamheten eller inte”, en person som har pensionerats kan anses inte längre vara aktiv i verksamheten men data om personen kan fortfarande behövas relaterat till en specifik verksamhet. De sistnämnda kraven anses dock inte lika avgörande för applikationens grundläggande funktionalitet som de första.

De viktigaste kraven har i Figur 1 markerats med en orange färg i stället för gul då de kan betecknas som avgörande för applikationen. För att numrera kraven används en autonumreringsfunktion inbyggd i verktyget Enterprise Architect.

### 2.1.2 Krav på personrelaterade data

Kraven i detta avsnitt kan sägas avgränsade till data om sådant en person har. Dessa data, som fortsättningsvis kommer att relateras till som personrelaterade data (Figur 2), är data som sannolikt kommer att förändras oftare än de grundläggande personattributen.

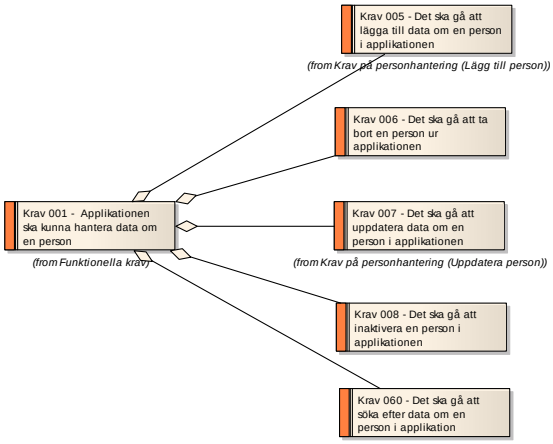



*Figur 2: Krav på personrelaterade data*

De viktigaste kraven i denna kategori utgörs av ”Krav 024 - Data om en person ska kunna innehålla adresser”, ”Krav 025 - Data om en person ska kunna innehålla telefonnummer”, ”Krav 026 - Data om en person ska kunna innehålla e-postadresser” och ”Krav 040 - Data om en person ska kunna innehålla uppgifter om anhöriga” där de tre första är gemensamma för alla applikationsdelarna medan det sista delas av anhörigdelen och den personaladministrativa delen av applikationen. Samtliga övriga krav är endast relaterade till den personaladministrativa delen av applikationen.

### 2.1.3 Krav på personhantering

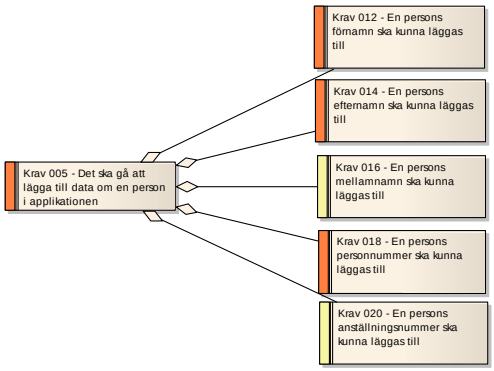
I kapitel 2.1.1 beskrivs krav på grundläggande personattribut med utgångspunkt i ”Krav 001 – Applikationen ska hunna hantera data om en person”. För att dessa data ska kunna hanteras följer ett antal andra krav relaterade till själva datahanteringen som också måste uppfyllas för att Krav 001 ska anses vara uppfyllt (Figur 3).




*Figur 3: Krav på personhantering*

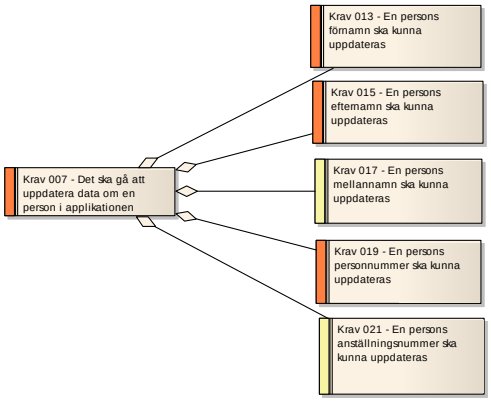
Dessa kan brytas ner till ”Krav 005 – Det ska gå att lägga till en person i applikationen”, ”Krav 006 – Det ska gå att ta bort en person ur applikationen”, ”Krav 007 – Det ska gå att uppdatera en person i applikationen” och ”Krav 008 – Det ska gå att inaktivera en person i applikationen” som är den funktion som i huvudsak ska användas istället för Krav 006, och slutligen ”Krav 060 Det ska gå att söka efter data om en person i applikationen” som är en förutsättning för Krav 007 och Krav 008.

Krav 005 kan sedan brytas ned i ett ytterligare krav för att kunna lägga till varje ytterligare data av typen grundläggande personattribut som behövs i applikationen, i Figur 4 utgörs dessa av Krav 012, Krav 014, Krav 016, Krav 018 och Krav 020 som kravställer möjligheten att lägga till förnamn, efternamn, mellannamn, personnummer och anställningsnummer.




*Figur 4: Krav för att hantera grundläggande personattribut när en person läggs till*

På samma sätt som för Krav 005 hanteras också Krav 007 (Figur 5) som bryts ned till Krav 013, Krav 015, Krav 017, Krav 019 och Krav 021 för att kunna uppdatera samma attribut som lades till i exemplet för Krav 005.



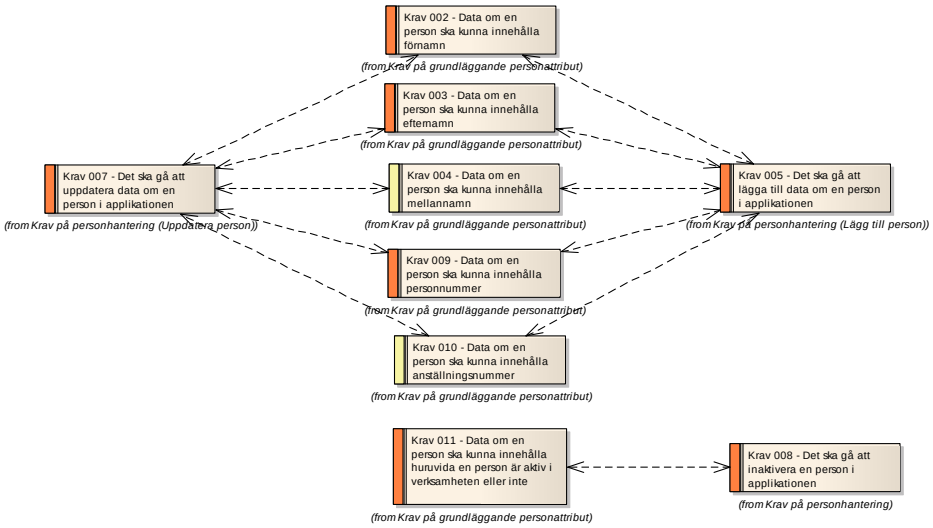

*Figur 5: Krav för att hantera uppdatering av grundläggande personattribut*

Krav 006 (Figur 3) bryts inte ned ytterligare då det handlar om att ta bort en komplett person ur applikationen, inte enskilda grundläggande personattribut. Om något data i de grundläggande personattribut som beskrivs i figuren ska tas bort hanteras detta genom kraven för att uppdatera. Däremot utökas Krav 006 av krav för att kunna ta bort andra personrelaterade data som till exempel adresser, telefonnummer och e-postadresser.

### 2.1.4 Beroenden mellan personattribut och personhantering

I Figur 6 visas beroenden för att lägga till en person i applikationen mellan krav som beskriver hantering av personer i systemet och krav som beskriver data om personer i

systemet. Det verkar i det närmaste självklart att om applikationen till exempel ska kunna innehålla data om en persons efternamn så måste detta kunna läggas till eller uppdateras och nästan lika självklart att om en person ska kunna inaktiveras så måste applikationen innehålla data om huruvida personen är aktiv eller inte (Krav 011).




*Figur 6: Några beroenden mellan grundläggande personattribut och personhantering*

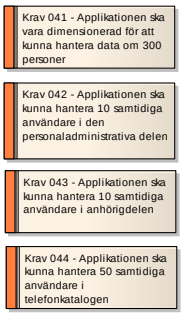
Det kan däremot sannolikt vara så att dessa beroenden uttryckta som i Figur 6 kan underlätta kommunikationen med beställaren genom att skapa en tydligare bild av hur applikationen slutligen kommer att bli. När beroenden mellan krav tydligt visas blir det också lättare att identifiera vilka krav som är viktigare än andra genom att dessa har många beroenden.

## 2.2 Icke-funktionella krav

Ett icke-funktionellt krav beskrivs av Wikipedia som ett krav som anger kriterier som kan användas för att bedöma driften av ett system, snarare än specifika beteenden [8]. I detta avsnitt beskrivs de icke-funktionella krav som formulerats under dialog med beställarens representant.

### 2.2.1 Krav på kapacitet

Krav på kapacitet (Figur 7) uttrycker de krav som ställs på applikationens förväntade kapacitet utifrån verksamhetens nuvarande bedömda behov.




*Figur 7: Krav på kapacitet*

”Krav 041 - Applikationen ska vara dimensionerad för att kunna hantera data om 300 personer” kravställer hur många anställda som applikationen är tänkt att hantera. ”Krav 042 - Applikationen ska kunna hantera 10 samtidiga användare i den personaladministrativa delen” kravställer hur många personaladministratörer som ska kunna arbeta samtidigt i applikationen. ”Krav 043 - Applikationen ska kunna hantera 10 samtidiga användare i anhörigdelen” kravställer hur många som ska kunna använda anhörigdelen samtidigt. Det sista kravet på applikationens kapacitet, ”Krav 044 - Applikationen ska kunna hantera 50 samtidiga användare i telefonkatalogen”, kravställer att åtminstone 50 användare ska kunna använda telefonkatalogen samtidigt.

### 2.2.2 Krav på plattform

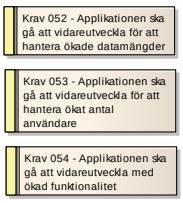
Krav på plattformen (Figur 8) uttrycker den målmiljö som applikationen ska användas i. De styr här vilka operativsystem som klientdelar och serverdelar kan installeras på. I detta fall styrs även valet av databashanterare och hur behörigheter ska regleras.


*Figur 8: Krav på plattform*

Utifrån kraven framgår att klienter för de olika applikationsdelarna ska köras på antingen Microsoft Windows XP eller Microsoft Windows 7, att serverdelar ska köras på antingen Microsoft Windows Server 2003 eller Microsoft Windows Server 2008 R2 och att databashanterare ska vara antingen Microsoft SQL Server 2005 eller Microsoft SQL Server 2008. Det framgår också att behörigheter till de olika applikationsdelarna ska regleras genom Microsoft Active Directory [9].

### 2.2.3 Krav på skalbarhet

Krav på skalbarhet (Figur 9) definierar i detta fall att de inledande kraven på kapacitet (2.2.1) kan komma att behöva revideras utifrån till exempel förändringar i beställarens organisation som kan innebära ett ökat antal användare och personer att hantera. Den applikationslösning som tas fram får då i sig inte innebära begränsningar utöver de som redan definierats genom krav på plattform (2.2.2).

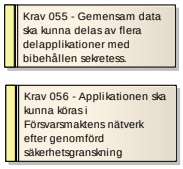



*Figur 9: Krav på skalbarhet*

Kraven på skalbarhet uttrycker att det totala antalet användare kan förväntas öka, att antalet personer som ska hanteras också kan förväntas öka och att det kan komma krav på ny och ökad funktionalitet.

### 2.2.4 Krav på säkerhet

Krav på säkerhet (Figur 10) definierar säkerheten i applikationen och kraven nedan kan utifrån vad som beskrivs här verka vara väldigt få, men så är inte fallet.




*Figur 10: Krav på säkerhet*

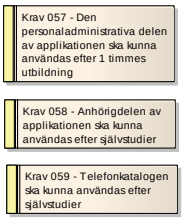
Kravet” Krav 056 - Applikationen ska kunna köras i Försvarsmaktens nätverk efter genomförd säkerhetsgranskning” innebär att hela applikationen tvingas in i den livscykelmodell, även kallad Auktorisationsprocessen, som används av Försvarsmakten

beskriven i Direktiv för Försvarsmaktens informationsteknikverksamhet[1] [10]. Denna process kan komma att generera nya krav, framförallt av icke-funktionell karaktär men dessa ligger som tidigare beskrivits (1.5) utanför ramen för detta arbete. En Auktorisation är en komplicerad process som tar lång tid att genomföra. Kritik har riktats mot Auktorisationsprocessen från bland annat Totalförsvarets forskningsinstitut, FOI, där Johan Bengtsson och Jonas Hallberg i en vetenskaplig rapport observerar att ”Auktorisations- och acrediteringsprocessen är anpassad för avgränsade mindre produkter, vilket leder till brist på helhetssyn och kontraproduktiva krav.” och ”Det är inte alltid den säkerhetsmässigt bästa lösningen som väljs utan den som kommer att godkännas i ackrediteringsprocessen. Ett typfall är att en äldre teknik som tidigare godkänts kan väljas istället för en nyare som ej tidigare har godkänts.” [11]. Eventuella problem med att passera processens ”nålsöga” har inte tagits hänsyn till i detta arbete utan den teknik och de metoder som ansets lämpligast för att på bästa sätt uppfylla beställerens behov har använts.

Det återstående kravet ”Krav 055 – Gemensam data ska kunna delas av flera delapplikationer med bibehållen sekretess.” är mer applikationsspecifikt och innebär att data om en person inte får ”läcka” mellan de olika applikationsdelarna då alla användare av applikationen inte är behöriga att ta del av all data. Det ska med andra ord till exempel inte gå att komma åt en persons personnummer genom telefonkatalogen.

### 2.2.5 Krav på användbarhet

Krav på användbarhet (Figur 11) beskriver hur enkel applikationen ska vara att använda ur ett användarperspektiv.



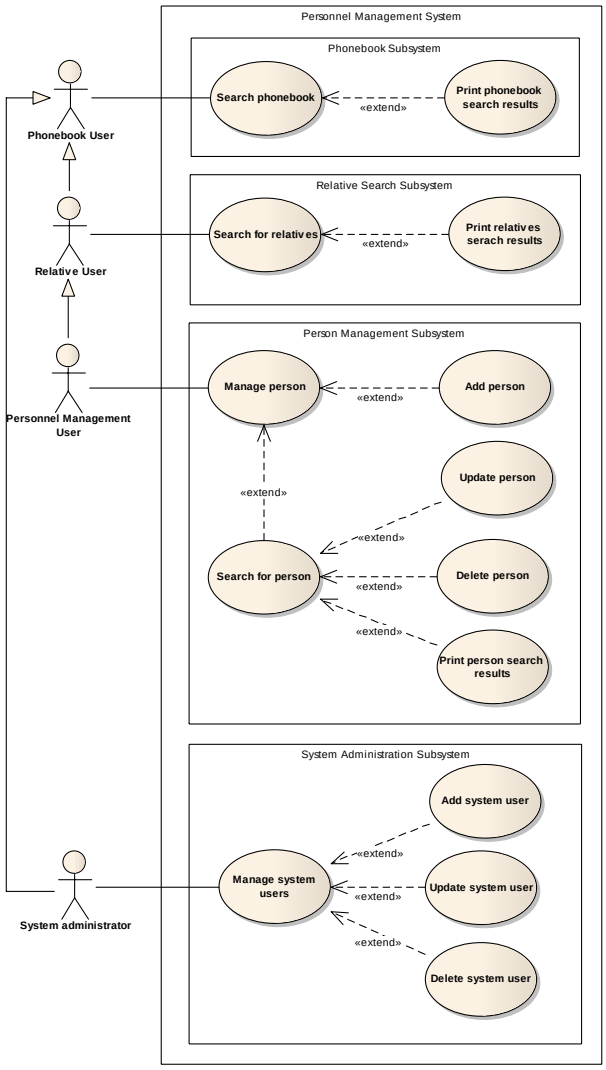

*Figur 11: Krav på användbarhet*

Applikationen ska vara så intuitiv och lätt att använda att det endast ska krävas en timmes utbildning för att kunna använda den personaladministrativa delen och endast självstudier för att hantera anhörigdel och telefonkatalog.

[1] Denna publikation benämns i dagligt tal som DIT 04.

# 3 Användningsfallsmodell

För att på ett tidigt stadium kunna skapa en övergipande bild av hur det nya systemet ska komma att se ut används en användningsfallsmodell. En användningsfallsmodell beskrivs med hjälp av modelleringsspråket Unified Modeling Language, UML [12]. Den innehåller användningsfall, Use-Cases, som används för att ”beskriva sekvenser av handlingar som aktörer (mänskliga eller icke-mänskliga entiteter som interagerar med systemet från utsidan) och systemet utför för att producera resultat av värde” [13]. Användningsfallsmodellen i detta kapitel kan sägas utgöra ett ytterligare förtydligande av de funktionella krav som beskrivs i kapitel 2.1. Modellen beskriver vilka aktörer som finns kopplade till systemet och vilka användningsfall dessa är kopplade till. Modellen (Figur 12) beskriver övergripande applikationen som kommer att kallas ”Personnel Management System” med dess ingående delapplikationer. I den personaladministrativa delen av applikationen, i modellen kallad ”Personnel Management Subsystem” hanteras data om personer. I anhörigdelen av applikationen, i modellen kallad ”Relative Search Subsystem” kan anhöriga till en angiven person sökas. I telefonkatalogen, i modellen kallad ”Phonebook Subsystem”, kan anställdas arbetsrelaterade telefonnummer, adresser och e-postadresser sökas. Slutligen i den systemadministrativa delen av applikationen, i modellen kallad ”System Administration Subsystem”, administreras behörigheter till de andra delarna av applikationen.

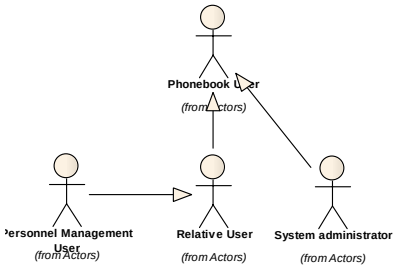



*Figur 12: Övergripande bild av systemet.*

## 3.1 Aktörer

Det finns två olika huvudtyper av aktörer i systemet som beskrivs i användningsfallsdiagrammet ovan (Figur 12). Dessa är Systemadministratör och Systemanvändare. Systemanvändaren kan delas in i tre undergrupper, användare av den personaladministrativa delen av applikationen, användare av anhörigdelen av applikationen och användare av telefonkatalogen (Figur 13). Pilarna mellan aktörerna i Figur 13 beskriver ett arv av egenskaper. För aktörernas del innebär det att användare av telefonkatalogen är de som har den lägsta behörigheten, en behörighet som systemadministratörer och övriga användare också har tillsammans med sina respektive utökade behörigheter. Högst behörighet i applikationen har användarna av den personaladministrativa delen som även har behörighet

att söka efter anhöriga och använda telefonkatalogen vilket är en förutsättning då de är de enda som tillför data till applikationen.




*Figur 13: Aktörer och deras relationer.*

### 3.1.1 Systemadministratör

Systemadministratören (Figur 13) är den typ av användare i systemet som administrerar de systemanvändare som ska ges behörighet till de olika applikationsdelarna. Denna typ av användare tillhör också gruppen användare som har rätt att använda telefonkatalogen. Det är lätt att anta att systemadministratören skulle kunna tillskansa sig fullständiga behörigheter till systemet men detta förhindras genom administrativa rutiner omhändertagna i det icke-funktionella ”Krav 056 - Applikationen ska kunna köras i Försvarsmaktens nätverk efter genomförd säkerhetsgranskning” (2.2.4) som ligger utanför ramarna för detta arbete.

### 3.1.2 Systemanvändare i applikationen

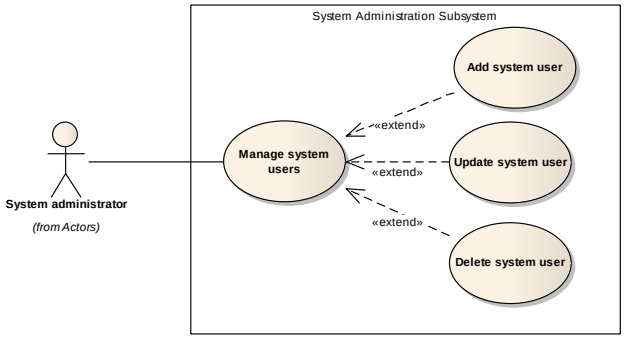
Systemanvändare i applikationen (Figur 13) är den typ av användare som ska använda någon av de tre olika applikationsdelarna i systemet. Systemanvändarna kan därför vara av tre olika typer, användare av den personaladministrativa delen av applikationen, användare av anhörigdelen av applikationen och användare av telefonkatalogen. En individ som är Systemadministratör kan bara vara användare av telefonkatalogen och ges därmed inte möjlighet att påverka data i applikationen vilket också definieras av ”Krav 056 - Applikationen ska kunna köras i Försvarsmaktens nätverk efter genomförd säkerhetsgranskning” .

Den användartyp med störst rättigheter vad avser dataåtkomst är användarna av den personaladministrativa delen av applikationen som har rätt att administrera data om de personer som avses lagras i systemet. Därefter i rättighetsordning kommer användare av

anhörigdelen av applikationen som är en typ av användare som har rätt att söka efter anhöriga till en specifik anställd person.
Den sista typen av användare är användare av telefonkatalogen, dessa har rätt att söka i telefonkatalogen. Till denna typ av användare räknas också användare tillhörande tidigare nämnda typer som också har rätt att använda telefonkatalogen.

## 3.2 Användningsfall för systemadministration

Användningsfallen för systemadministration (Figur 14) hanterar de användningsfall som har med hantering av andra användartyper i applikationen att göra.




*Figur 14: Användningsfall för systemadministration.*

### 3.2.1 Manage system users

”Manage system users” är ett generellt användningsfall för att hantera en systemanvändare i den del av systemet som är till för systemadministration (Figur 14). Användningsfallet utökas av tre användningsfall, ”Add system user” (3.2.2), för att lägga till en systemanvändare, ”Update system user” (3.2.3) för att uppdatera en systemanvändare och ”Delete system user” (3.2.4) för att ta bort en systemanvändare. Samtliga dessa användningsfall är därför något som endast aktören Systemadministratör (3.1.1) har rätt att använda.

### 3.2.2 Add system user

Användningsfallet ”Add system user” för att lägga till en systemanvändare är det första av de användningsfall som utökar användningsfallet ”Manage system users” (3.2.1) och innebär att en användare som tidigare inte varit behörig nu ges behörighet att använda någon eller några av de tre applikationsdelarna.

### 3.2.3 Update system user

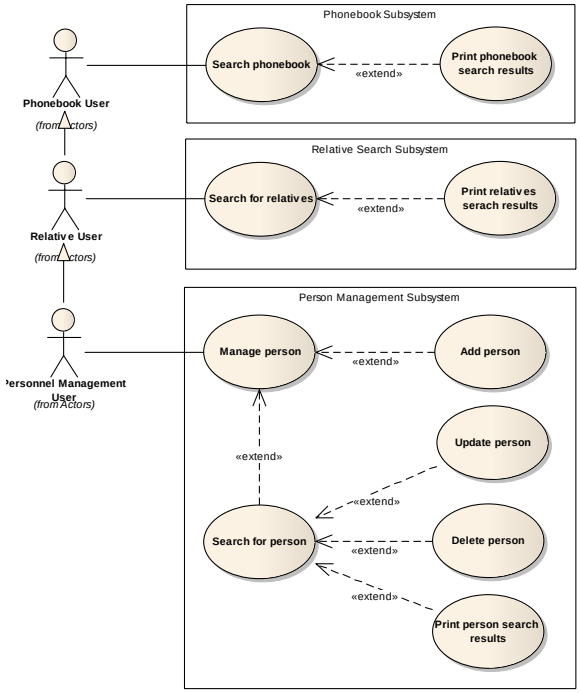
Användningsfallet ”Update system user” för att uppdatera en systemanvändare är det andra av de användningsfall som utökar användningsfallet ”Manage system users” (3.2.1) och innebär att en användares behörigheter för användning av någon av de tre applikationsdelarna förändras. Denna förändring består endast av att tillgången till applikationsdelarna förändras, inte att all behörighet till applikationen tas bort då detta utgörs av ett eget användningsfall.

### 3.2.4 Delete system user

Användningsfallet ”Delete system user” för att ta bort en systemanvändare är det tredje och sista av de användningsfall som utökar ”Manage system users” (3.2.1) och innebär att en användares behörigheter att använda samtliga av de tre applikationsdelarna tas bort. Användaren kommer således efter borttagen behörighet inte att kunna nå någon applikationsdel.

## 3.3 Primära användningsfall för systemanvändning

De primära användningsfallen för systemanvändning (Figur 15) beskriver endast de användningsfall som är relaterade till hantering av de grundläggande personattribut som utgörs av förnamn, efternamn, mellannamn, personnummer, och huruvida personen är i aktiv tjänst eller inte. Övriga data hanteras via de sekundära användningsfallen i avsnitt 3.4.




*Figur 15: Användningsfall för systemanvändning.*

### 3.3.1 Manage person

”Manage person” ett generellt användningsfall för att hantera personer i den personaladministrativa delen av applikationen (Figur 15). Detta användningsfall utökas av användningsfallen ”Add person” (3.3.2) och ” Search for person” (3.3.3) där det sistnämnda i sin tur utökas av ”Update person” (3.3.4), ”Delete person” (3.3.5) och ”Print person search results” (3.3.6). Gemensamt för dessa användningsfall är att de bara kan användas av användare av den personaladministrativa delen av applikationen som också är den typ av användare som har de största behörigheterna i applikationen relaterat till den mängd data de har tillgång till.

### 3.3.2 Add person

Användningsfallet ”Add person” utökar det generella användningsfallet ”Manage person” (3.3.1) och innebär att en person läggs till i applikationen. Detta användningsfall innebär bara skapande av de attribut som är direkt kopplade till en person det vill säga personnummer, förnamn, mellannamn, efternamn och anställningsnummer. Dessa attribut har också utökats med kön, en personspecifik anteckning och en unik identifierare för personen som genereras av applikationen och inte kan förändras. För att resterande data ska kunna tillföras en person så utökas detta användningsfall av sekundära användningsfall för att bland annat hantera adresser, hantera telefonnummer, hantera e-postadresser, hantera körkort och så vidare. Dessa användningsfall beskrivs mer ingående i avsnitt 3.4.

### 3.3.3 Search for person

Användningsfallet ”Search for person” tillhör den personaladministrativa delen av applikationen. Det utökar det generella användningsfallet ”Manage person” (3.3.1) och föregår alltid användningsfallen ”Update person” (3.3.4), ”Delete person” (3.3.5) och ”Print person search results” (3.3.6). Användningsfallet möjliggör sökning efter personer i applikationen utifrån ett angivet sökord där sedan resultatet presenteras som en lista i vilken önskad person kan väljas för vidare behandling. Sökorden som kan anges är hela eller del av eftersökt individs personnummer, förnamn, mellannamn eller efternamn.

### 3.3.4 Update person

Användningsfallet ”Update person” utökar användningsfallet ” Search for person” (3.3.3) och innebär att en redan befintlig person i applikationen uppdateras efter att ha modifierats. Uppdateringen rör bara de attribut som är direkt kopplade till en person, precis som för användningsfallet ”Add person” (3.3.2), det vill säga ändringar vad gäller personnummer,

förnamn, mellannamn, efternamn, anställningsnummer, kön och en personspecifik anteckning. Den unika identifieraren kan inte ändras. Precis som för ”Add person” utökas även detta användningsfall av samma sekundära användningsfall för att bland annat hantera adresser, hantera telefonnummer, hantera e-postadresser, hantera körkort och så vidare. Dessa användningsfall beskrivs mer ingående i kapitel 3.4.

### 3.3.5 Delete person

Användningsfallet ”Delete person” utökar användningsfallet ”Search for person” (3.3.3) och innebär att all data om en person som inte delas med någon annan person raderas ur applikationen. Detta är tänkt att ske ytterst sällan då tanken är att personen i stället görs inaktiv något som kan göras genom användningsfallet ”Update person” (3.3.4) och på så sätt ändå är tillgänglig vid sökningar i applikationen. Det underlättar också till exempel förfarandet vid återanställning av en tidigare anställd person.

### 3.3.6 Print person search results

Användningsfallet ” Print person search results” som också utökar användningsfallet ”Search for person” (3.3.3) innebär att en rapport med all data om vald person, inklusive adresser, telefonnummer och så vidare formateras till ett utskriftsvänligt format för att kunna skrivas ut på skrivare eller till exempelvis PDF-format.

### 3.3.7 Search for relatives

Detta användningsfall är det viktigaste av de två användningsfall som tillhör anhörigdelen (Figur 15) av applikationen. Denna del av applikationen är den del som kräver näst högst behörighet i systemet relaterat till den mängd data användarna kommer i kontakt med.
Användningsfallet innebär genomförandet av en sökning efter person där det sökord som anges är hela eller del av eftersökt individs personnummer, förnamn, mellannamn eller efternamn. Sökningen ska förutom data om den eftersökte personen och dennes adresser också resultera i en lista på dennes anhöriga, vilken relation de har till sökobjektet och deras i applikationen angivna adresser och telefonnummer.

### 3.3.8 Print relatives serach results

Det andra användningsfallet av de två tillhörande anhörigdelen är ”Print relatives search results” är en utökning av användningsfallet ”Search for relatives” (3.3.7) vilket innebär att sökresultatet från ” Search for relatives” ska formateras till ett utskriftsvänligt format för att kunna skrivas ut på skrivare eller till exempelvis PDF-format.

### 3.3.9 Search phonebook

Telefonkatalogen ska vara åtkomlig för en betydligt större mängd personal än de första två delarna men medger också åtkomst till en betydligt mindre mängd data. Användningsfallet ” Search phonebook” är det viktigaste av de två användningsfall som är relaterade till telefonkatalogen och innebär genomförandet av en sökning efter person där hela eller del av personens förnamn eller efternamn används som sökord. Sökningen ska resultera i en lista med anställda med namn, organisationstillhörighet, telefonnummer till fast arbetstelefon, mobil arbetstelefon, e-post för arbete, adress till arbetsplats och eventuellt foto. Sökning ska också kunna genomföras på organisationsenhet för att på samma sätt som beskrivits ovan kunna lista de som tillhör enheten.

### 3.3.10 Print phonebook search results

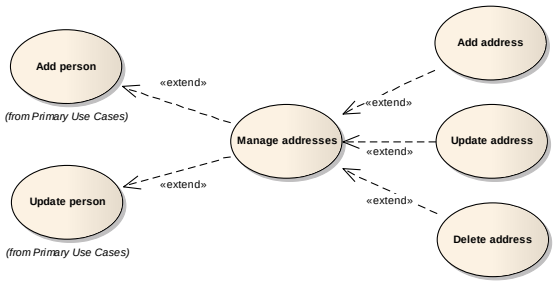
Användningsfallet **”** Print phonebook search results” är det andra av de två användningsfall som ingår i telefonkatalogen. Det utökar användningsfallet ” Search phonebook” (3.3.9) och innebär att sökresultatet från detta formateras till ett utskriftsvänligt format för att kunna skrivas ut på skrivare eller till exempelvis PDF format.

## 3.4 Sekundära användningsfall för systemanvändning

De sekundära användningsfallen används för att beskriva den hantering av data som inte är direkt knuten till en person så som adresser, telefonnummer, e-postadresser och så vidare.
Att de anges som sekundära innebär inte att de är mindre viktiga utan snarare att de utökar de primära användningsfallen ytterligare. Uppdelningen mellan primära och sekundära användningsfall ska således betraktas som ett försök att göra hela användningsfallsmodellen mer överskådlig. Nedan redovisas användningsfall för adresser, telefonnummer, e-postadresser då dessa används av alla delar av applikationen.

### 3.4.1 Manage Address

Det generella användningsfallet ”Manage adress” (Figur 16) utökar användningsfallen ”Add person” (3.3.2) och ”Update person” (3.3.4) och möjliggör hanterandet av en persons olika adresser.




*Figur 16: Användningsfall för hantering av adresser*

Detta kräver naturligtvis en utökning med användningsfallen ”Add address” som lägger till en unikt identifierad adress i applikationen, ”Update address” som möjliggör uppdatering av en befintlig adress och ”Delete address” som raderar adressen permanent ur applikationen. Adresserna är kopplade till en unik person så om två personer skulle bo på samma adress och den ena adressen av någon anledning ska raderas så påverkas inte den andra. Adresser kan vara till arbetet eller privata. Det är endast arbetsrelaterade adresser som ska visas i telefonkatalogen.

### 3.4.2 Manage phonenumber

”Manage phonenumber” är ett generellt användningsfall för hantering av telefonnummer. Det utökas av användningsfallen ”Add phonenumber” som möjliggör att ett nytt telefonnummer kan läggas till, ”Update phonenumber” som möjliggör att ett telefonnummer kan uppdateras och ”Delete phonenumber” som innebär att telefonnumret raderas permanent ur applikation. Alla telefonnummer är unika och kopplade till en individ så samma telefonnummer kan, precis som i fallet för adresser, förekomma flera gånger i applikationen. ”Manage phonenumber” utökar användningsfallen ”Add person” (3.3.2) och ”Update person” (3.3.4). Telefonnummer kan vara till arbetet eller privata och av typen vanlig telefon, mobiltelefon, personsökare eller fax. Personsökare och faxnummer är inte privata. Det är bara arbetsrelaterade telefonnummer som kommer att visas i telefonkatalogen.

### 3.4.3 Manage emailaddress

Precis som för ”Manage address” och ”Manage phonenumber” så utökar det generella användningsfallet ”manage emailaddress” användningsfallen ”Add person” (3.3.2) och ”Update person” (3.3.4). Användningsfallet utökas av användningsfallen ”Add emailaddress”, Update emailaddress” och ”Delete emailaddress” som ger möjlighet att lägga till en unik e-postadress, uppdatera en befintlig e-postadress eller permanent radera en e-postadress. Alla e-

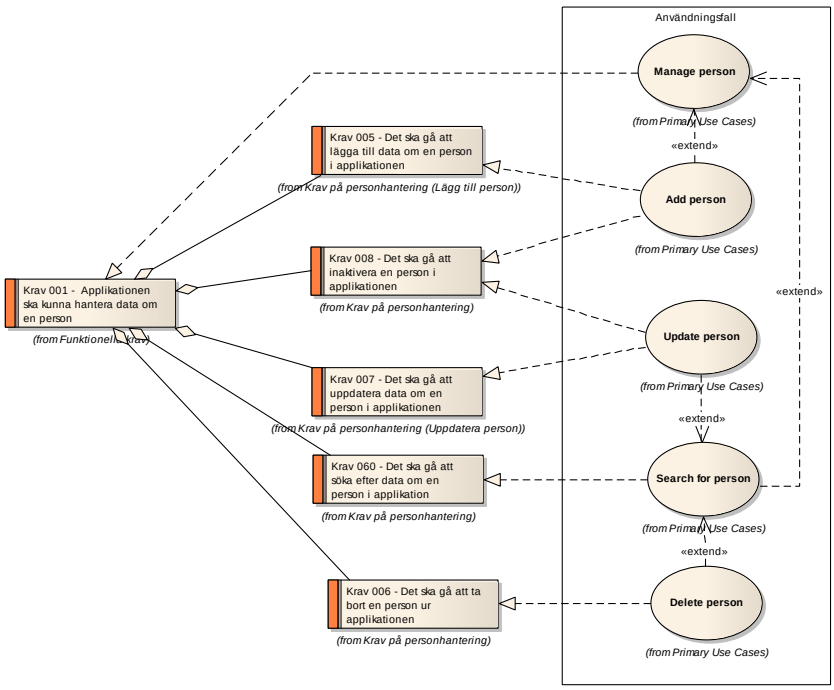
postadresser betraktas som unika. E-postadresser kan vara till arbetet eller privata och det är bara arbetsrelaterade e-postadresser som kommer att visas i telefonkatalogen.

## 3.5 Spårbarhetsmodell

För att säkerställa att alla krav är omhändertagna skapas en enkel spårbarhetsmodell mellan krav och användningsfall. Den visar vilket eller vilka användningsfall som används för att realisera vilket krav.

### 3.5.1 Hantering av person

Användningsfallet ”Manage person” (3.3.1) används för att realisera ”Krav 001 - Applikationen ska kunna hantera data om en person” (2.1.3) och hur detta går till beskrivs av Figur 17.




*Figur 17: Spårbarhetsmodell för hantering av person*

Som tidigare beskrivits utökas användningsfallet ”Manage person” av användningsfallen ”Add person” (3.3.2) och ”Search for person” (3.3.3). Dessa användningsfall realiserar i sin tur ”Krav 005 – Lägg till person” respektive ”Krav 060 – Sök person” som är två av de krav som måste vara uppfyllda för att ”Krav 001 - Applikationen ska kunna hantera data om en person” ska anses uppfyllt. Användningsfallet ”Add person” är också med och realiserar ” Krav 008 - Det ska gå att inaktivera en person i applikationen” (2.1.3) tillsammans med användningsfallet ”Update person” (3.3.4) då en person ska kunna läggas till som inaktiv

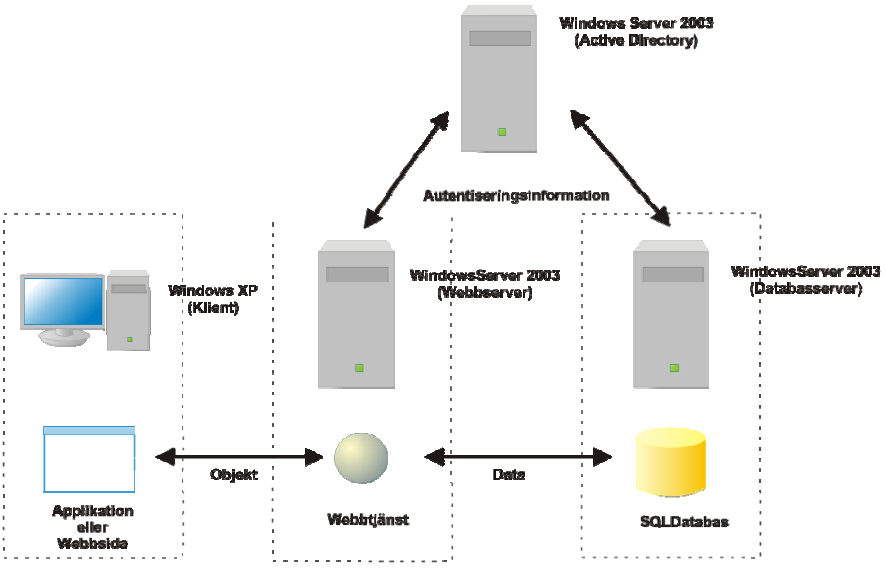
direkt vid skapandet och att detta ska kunna uppdateras, något som skulle kunna vara fallet med en person som först läggs till som anhörig men som senare anställs. För att en person ska kunna uppdateras måste data om personen först hittas i applikationen. Detta görs med användningsfallet ”Search for person” (3.3.3) som därmed realiserar ” Krav 060 - Det ska gå att söka efter data om en person i applikation” (2.1.3). Användningsfallet ” Search for person” föregår också alltid användningsfallet ”Delete person” (3.3.5) som realiserar ” Krav 006 - Det ska gå att ta bort en person ur applikationen” (2.1.3).

# 4 Analys

Med utgångspunkt i tidigare beskrivna krav (0) och användningsfall (0) genomförs en analys uppdelad i fem delar för den personaladministrativa delen av applikationen. Den första delen beskriver analys av hur applikationen ska byggas upp för att realisera tidigare beskrivna ickefunktionella krav (2.2). Därefter beskrivs i den andra delen av analysen hur de data som till stora delar är gemensamma för applikationens olika delar (4.2) ska knytas samman. Den tredje delen beskriver nödvändiga komponenter för att skapa en klient (4.3). Den fjärde delen beskriver nödvändiga komponenter för att skapa servern (4.4) och den femte och sista delen av analysen (4.5) beskriver hur användningsfallet ”Manage person” (3.3.1) realiseras utifrån beskrivningarna i avsnitt 4.2, 4.3 och 4.4.

## 4.1 Applikationens uppbyggnad

För att tillgodose de icke-funktionella kraven för plattform så dimensioneras systemet för de mest gränssättande av kraven. Applikationen delas upp i en klientdel, en serverdel och en databasdel, den använder också ett befintligt Active Directory för att hantera autentisering och behörigheter enligt ”Krav 051 - Behörigheter till applikationen ska kunna hanteras via Microsoft Active Directory” (2.2.2). En principskiss för systemets uppbyggnad visas i Figur 18.




*Figur 18: Principskiss över systemets uppbyggnad.*

Klientens operativsystem blir Windows XP enligt ” Krav 045 - Applikationens klientdel ska kunna köras på Microsoft Windows XP” (2.2.2) eftersom den då med stor sannolikhet utan

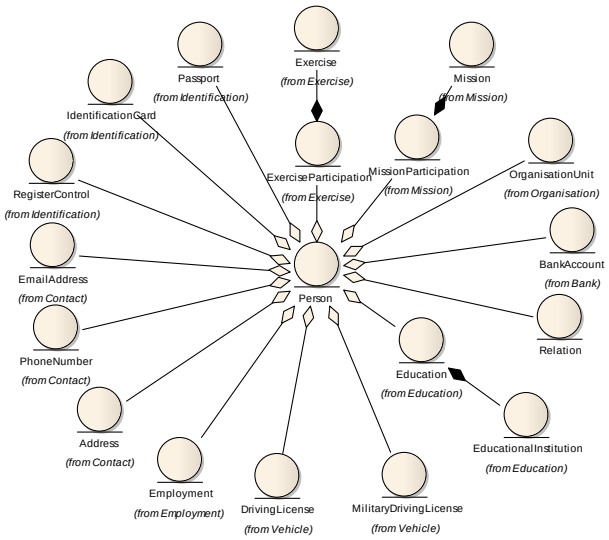
modifikationer också kan köras i Windows 7 enligt ”Krav 046 - Applikationens klientdel ska kunna köras på Microsoft Windows 7” (2.2.2). Detta antagande måste naturligtvis verifieras. Klienten för den personaladministrativa delen av applikationen kommer att utvecklas som en Windows Forms-applikation [14] medan anhörigdelen och telefonkatalogen blir webbsidor som hanteras via klientens webbläsare (Figur 18).

Samma resonemang antas för val av serverns operativsystem det vill säga Microsoft Windows Server 2003 används enligt ” Krav 047 - Applikationens serverdel ska kunna köras på Microsoft Windows Server 2003” (2.2.2) för att sannolikt utan modifikationer kunna flytta serverdelen till Microsoft Windows Server 2008 R2 enligt ” Krav 048 - Applikationens serverdel ska kunna köras på Microsoft Windows Server 2008 R2” (2.2.2). Applikationens serverdel kommer för att enkelt kunna kommunicera med de olika klientdelarna och med databasen utvecklas som en webbtjänst [15] som körs på en webbserver, i detta fall Microsoft Internet Information Server (IIS), i vilken också webbsidorna för anhörigdelen och telefonkatalogen ligger (Figur 18).

Databasen kommer att skapas i Microsoft SQL Server 2005 enligt ”Krav 049 - Applikationens data ska kunna lagras i Microsoft SQL Server 2005” (2.2.2). och kan enkelt flyttas till Microsoft SQL Server 2008 enligt ” Krav 050 - Applikationens data ska kunna lagras i Microsoft SQL Server 2008” (2.2.2).
Med vald systemlösning antas också samtliga kapacitetskrav (2.2.1) och krav på skalbarhet (2.2.3) vara uppfyllda.

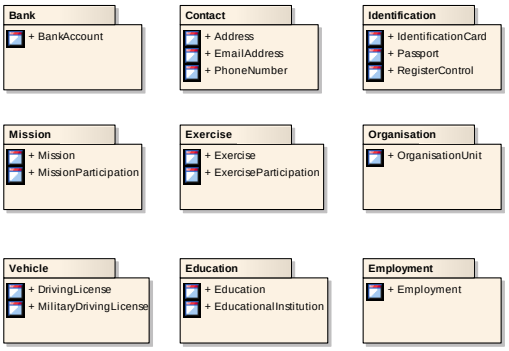
## 4.2 Gemensamma analysklasser

Den viktigaste i de tre applikationsdelarna är hur data om en person ska hanteras. Detta blev därför den viktigaste delen att analysera. Resterande del av detta avsnitt fokuserar på hur data för en person (Figur 19) hanteras. Utgångspunkten för analysen blir det som i kraven benämns grundläggande personattribut. Dessa samlas som attribut i analysklassen Person. Dessa data kan förväntas ha en långsammare förändringstakt och åldras sannolikt inte lika fort som till exempel data om adresser och telefonnummer. Data så som adresser och telefonnummer hanteras istället som egna analysklasser och lagras i kollektioner i analysklassen Person.




*Figur 19: Analysklasser för att beskriva en Person.*

För underlätta hanteringen av analysklasser [12] tillhörande en person skapades ett antal analyspaket [12] i vilka klasserna inordnades (Figur 20).




*Figur 20: Analyspaket tillhörande Person.*

Gemensamt för alla analysklasser utom OrganisationUnit, Education, EducationalInstitution, Exercise och Mission är att de är relaterade till en specifik person. För att exemplifiera detta kan man tänka sig att även om en anhörig delade samma adress så är detta tänkt att denna adress hanteras som två olika adresser medans flera personer kan tillhöra samma organisations enhet och flera personer kan ha gått samma kurs samtidigt eller vid olika tillfällen.

### 4.2.1 Analysklassen Person

Klassen för person (Figur 19) är den enskilt viktigaste klassen i applikationen. Den innehåller data om en persons unika identifierare i applikationen, förnamn, mellannamn, efternamn,

personnummer, om personen är aktiv i verksamheten och en anteckning. Förutom detta så innehåller klassen kollektioner av bland annat adresser, telefonnummer, e-postadresser, id-kort, pass, genomförda registerkontroller, körkort, militära förarbevis, utbildningar, militära övningar och missioner, eventuella andra anställningar, organisationstillhörighet, bankkonton och angivna relationer till andra personer.

### 4.2.2 Analysklassen Address

Analysklassen Address (Figur 19) tillhör analyspaketet Contact och beskriver en adress (Figur 20) specifikt relaterad till en person. Adressklassen innehåller en i applikationen unik identifierare för att göra adressen unik. Den innehåller också gatunamn, ortsnamn, postnummer och vilken typ av adress det rör sig om, huruvida det är en arbetsrelaterad adress eller en hemadress och om den ska visas i telefonkatalogsdelen av applikationen. Visningsvalet gäller bara för de adresser som är arbetsrelaterade. Utifrån detta framgår att det därför bara kommer att vara de personer som har behörighet till den personaladministrativa delen av applikationen som avgör vad som kommer att visas i telefonkatalogsdelen av applikationen. Alla adresser kommer däremot att visas i anhörigdelen av applikationen.

### 4.2.3 Analysklassen Phonenumber

Analysklassen PhoneNumber (Figur 19) tillhör analyspaketet Contact och beskriver telefonnummer relaterat till en specifik person. Numret kan vara hem eller till arbetet, den kan också vara till mobiltelefon, vanlig telefon, fax, eller personsökare. Liksom analysklassen Address (4.2.2) så innehåller också PhoneNumber huruvida numret ska visas i telefonkatalogsdelen av applikationen och det är under förutsättning att det är arbetsrelaterat.

### 4.2.4 Analysklassen EmailAddress

Analysklassen EmailAddress (Figur 19) tillhör analyspaketet Contact och innehåller en e-postadress relaterad till en specifik användare. Den innehåller också om det är en privat adress eller en arbetsrelaterad adress. Precis som med analysklasserna Address (4.2.2) och PhoneNumber (4.2.3) så innehåller också EmailAddress huruvida numret ska visas i telefonkatalogsdelen av applikationen och precis som för de andra klasserna så är det under förutsättning att den är arbetsrelaterad.

### 4.2.5 Analysklassen Passport

Analysklassen Passport (Figur 19) tillhör analyspaketet Identification och används för att beskriva ett pass tillhörande en specifik person. Den innehåller personens förnamn,

mellannamn, efternamn, passnummer och slutdatum för passets giltighet. Namnuppgifterna fylls i automatiskt när objektet skapas och är till för att hantera tillfälliga olikheter mellan grundläggande personattribut och pass som kan uppstå vid till exempel namnbyte.

### 4.2.6 Analysklassen IdentificationCard

Analysklassen IdentificationCard (Figur 19) tillhör analyspaketet Identification och används för att beskriva ett ID-kort för en specifik person. Den innehåller personens förnamn, mellannamn, efternamn, ID-kortnummer och slutdatum för kortets giltighet. Namnuppgifterna fylls, precis som för passet, i automatiskt när objektet skapas något som är till för att hantera tillfälliga olikheter mellan grundläggande personattribut och ID-kortsuppgifter som kan uppstå vid till exempel namnbyte.

### 4.2.7 Analysklassen RegisterControl

Analysklassen RegisterControl tillhör analyspaketet Identification och används för att lagra ett tillfälle när en registerkontroll gjorts för en specifik person. Den innehåller registerkontrollklass, datum för senaste uppdatering, slutdatum för kontrollens giltighet och kontrollorsak.

### 4.2.8 Analysklassen Relation

Analysklassen Relation används för att beskriva en riktning av en relation mellan två personer. Klassen innehåller två personers unika identifierare och en relationsbeskrivning. Innehållet ska utläsas som att person ett har en relation till person två. Relationen är enkelriktad då samma relation inte existerar åt andra hållet. Relationsbeskrivningen ska utläsas som person tvås roll i förhållande till person ett.

### 4.2.9 Analysklassen OrganisationUnit

Analysklassen OrganisationUnit (Figur 19) tillhör analyspaketet Organisation och används för att beskriva en organisationsenhet. Klassen innehåller enhetens namn, och vilken annan organisationsenhet den lyder under.

### 4.2.10 Analysklassen Education

Analysklassen Education (Figur 19) tillhör analyspaketet Education och används för att beskriva en utbildning. Klassen innehåller utbildningens namn, vilken typ av utbildning det handlar om, till exempel om det är en kurs eller en certifiering, på vilken utbildningsinstitution (4.2.11) den genomfördes, hur många högskolepoäng eller motsvarande den ger, startdatum, slutdatum och om den har ett bäst före datum.

### 4.2.11 Analysklassen EducationalInstitution

Analysklassen EducationalInstitution (Figur 19) tillhör analyspaketet Education och beskriver en utbildningsinstitution som till exempel ett universitet eller ett utbildningsföretag. En utbildning som beskrivs av analysklassen Education (4.2.10) är alltid kopplad till en utbildningsinstitution. Varje institution innehåller data om institutionens namn, telefonnummer, gatuadress, postnummer, stad, land, postboxnummer, e-postadress och en unik identifierare. Klassen är för närvarande inte tänkt att utnyttja de befintliga analysklasserna för adress, telefonnummer eller e-postadress men detta kan komma att ändras senare.

### 4.2.12 Analysklassen Exercise

Analysklassen Exercise (Figur 19) tillhör analyspaketet Exercise och beskriver en militär övning. Den är avsedd att generellt beskriva en övning och innehåller övningens namn, plats startdatum och slutdatum. Då det är ganska vanligt att deltagare i en övning inte deltar under hela övningsperioden så hanteras enskilda individers deltagande i övningen via analysklassen ExerciseParticipation (4.2.13).

### 4.2.13 Analysklassen ExerciseParticipation

Analysklassen ExerciseParticipation (Figur 19) tillhör analyspaketet Exercise och beskriver en specifik persons deltagande i en övning. Den innehåller därför personens unika identifierare, från vilket datum personen ingick i övningen, från vilket datum personen utgick ur övningen och vilken övning det var (4.2.12).

### 4.2.14 Analysklassen Mission

Analysklassen Mission (Figur 19) tillhör analyspaketet Mission och beskriver en militär insats. Den är avsedd att generellt beskriva insatsen och innehåller insatsens namn startdatum, slutdatum och plats. Den är i det närmaste identisk med klassen Exercise (4.2.12) och det skulle ha gått att använda en klass för att beskriva dem båda men valet att skapa en egen klass beror på att klassen kan behöva kompletteras med ytterligare data. Precis som för Exercise så finns också en klass för att beskriva en specifik persons deltagande i insatsen (4.2.15).

### 4.2.15 Analysklassen MissionParticipation

Analysklassen MissionParticipation (Figur 19) tillhör analyspaketet Mission och beskriver en specifik persons deltagande i en insats. Den innehåller därför personens unika identifierare,

från vilket datum personen ingick i insatsen, från vilket datum personen utgick ur insatsen och vilken insats det var (4.2.14).

## 4.3 Analysklasser för klient

Analysklasserna för att beskriva klienten handlar främst om att beskriva användargränssnitt för att fylla de olika gemensamma analysklasserna (4.2) med data. Gränssnitten hålls samman av analysklassen MainForm (4.3.2) som utgör den personaladministrativa delen av applikationens huvudgränssnitt. Nästan varje objekt som ska fyllas med data har ett eget korresponderande gränssnitt. Förutom dessa gränssnitt finns även analysklasser för kommunikation med servern. Detta kommer som tidigare beskrivits att ske via en webbtjänst.
I resterande del av detta avsnitt presenteras kort de två viktigaste analysklasserna för klienten.

### 4.3.1 Analysklassen WebServiceAbstraction

Analysklassen WebServiceAbstraction utgör kommunikationsgränssnittet mot webbtjänsten. Den sköter eventuell formatomvandling och sköter anrop mot webbtjänstens metoder.

### 4.3.2 Anlysklassen MainForm

Analysklassen MainForm är den sammanhållande komponenten i klienten. Det är genom den som alla andra gränssnitt kan nås. MainForm tillhandahåller gränssnitt för att kunna söka efter personer för att sedan möjliggöra presentation av all data om den person som valts.

## 4.4 Analysklasser för server

Analysklasser för server består av två klasser, en som utgör webbtjänsten som klienter kommunicerar med och en som utgör dataabstraktionslagret som kommunicerar med databasen.

### 4.4.1 Analysklassen WebService

Analysklassen WebService beskriver webbtjänsten och hanterar serverdelen av kommunikationen. Klassen kan sägas agera som en Proxy mot analysklassen DataAbstraction (4.4.2) då det enda som genomförs är att kontrollera att användaren som försöker komma åt en specifik metod är behörig att göra det för att om så är fallet anropa motsvarande metod i DataAbstraction.

### 4.4.2 Analysklassen DataAbstraction

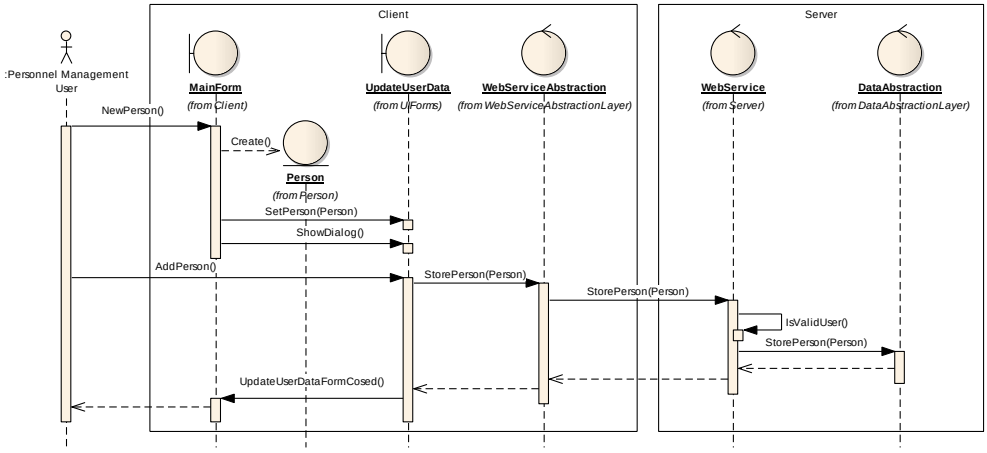
Analysklassen DataAbstraction hanterar kommunikationen mot databasen. Klassen hanterar hämtning av data och sammansättning av dessa till objekt från databasen och lagring av objekt till databasen.

## 4.5 Realisering av Användningsfall

Med hjälp av de analysklasser som beskrivs för gemensamma objekt (4.2), för klient (4.3) och för server (4.4) kan de användningsfall som beskrivs för den personaladministrativa delen av applikationen, det vill säga ”Add person” (3.3.2), ”Search for person” (3.3.3), ”Update person” (3.3.4) och ”Delete person” (3.3.5), realiseras.

### 4.5.1 Lägg till person

Användningsfallet “Add person” (3.3.2) beskriver hur en person läggs till i applikationen. Detta användningsfall realiseras (Figur 21) genom att användaren för den personaladministrativa delen av applikationen (3.1.2) via Klientens huvudformulär, MainForm (4.3.2), väljer att skapa en ny person. MainForm skapar då ett nytt personobjekt och öppnar detta för editering genom att visa objektet i UpdateUserData (4.3) som är gränssnittet för att lägga till och uppdatera data tillhörande de grundläggande personattributen. När användaren är nöjd med det data denne tillfört kan personobjektet sparas. Detta görs genom att UpdateUserData anropar WebServiceAbstraction (4.3.1) varifrån objektet lämnar klienten och skickas vidare till Serverns WebService (4.4.1). En kontroll av att användaren är behörig utförs innan personobjektet skickas till DataAbstraction som ser till att dess data lagras i databasen.

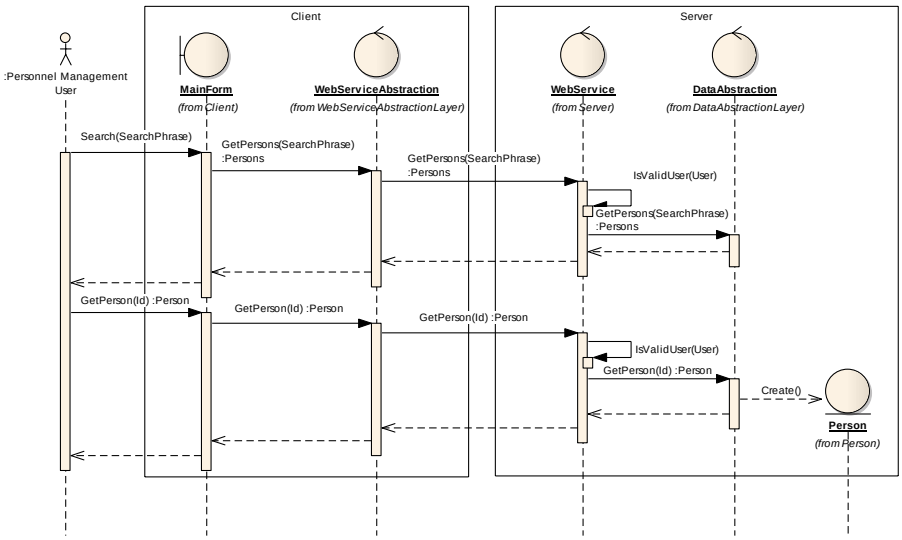



*Figur 21: Realisering av användningsfallet ”Lägg till person”.*

Om lagringen lyckas stängs UpdateUserData och MainForm uppdaterar gränssnittet för att visa den nyskapade personens data om dennes grundläggande personattribut hämtade ur personobjektet.

### 4.5.2 Sök person

Som beskrivs i användningsfallet ”Search for person” (3.3.3) så föregår en sökning av en person användningsfallen ”Update person” (3.3.4) och ”Delete person” (3.3.5). Användningsfallet realiseras (Figur 22) genom att användaren av den personaladministrativa delen av applikationen (3.1.2) via klientens sökfunktion i MainForm (4.3.2) skriver in ett sökord som passar in på den eftersökta personen. Sökordet kan vara del av förnamnet, mellannamnet, efternamnet eller personnumret.




*Figur 22: Realisering av användningsfallet ”Sök person”.*

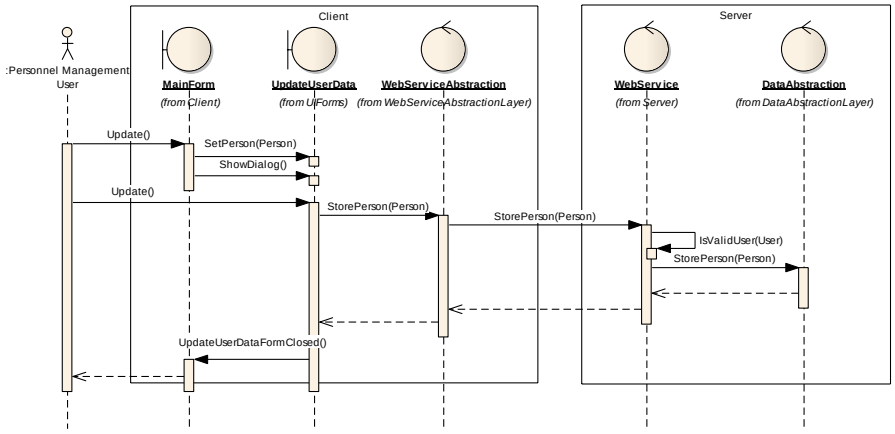
MainForm skickar sökordet till WebServiceAbstraction (4.3.1) varifrån sökordet lämnar klienten och skickas vidare till serverns WebService (4.4.1). En kontroll av att användaren är behörig utförs innan sökordet skickas vidare till DataAbstraction (4.4.2). DataAbstraction gör en sökning i databasen utifrån sökordet och sammanställer en lista på personer som uppfyller sökkriteriet. Listan som returneras innehåller bara de grundläggande personattributen för att hålla nere mängden kommunicerad data. Listan returneras via Serverns Webservice till klientens WebServiceAbstraction och sedan till MainForm i vilket sökresultatet presenteras.

Om användaren inte hittar vad denne söker i resultatet kan naturligtvis sökningen återupprepas men om det han söker finns kan detta objekt väljas genom ett dubbelklick med musen. Ett anrop för att hämta önskat objekt går då från MainForm via WebServiceAbstraction och WebService, som gör en ny kontroll av användarens behörighet,

till DataAbstraction som skapar ett nytt personobjekt och klär på detta med all data om personen som kan hittas i databasen. Personobjektet returneras sedan tillbaka samma väg som anropet för att hämta personen kom. Den fullständiga personen presenteras sedan i MainForm där det finns möjlighet att lägga till, uppdatera eller ta bort de olika delarna av data som utgör beskrivningen av personen.

### 4.5.3 Uppdatera person

Att uppdatera data om en person föregås alltid av användningsfallet ”Search for person” (3.3.3) som realiseras i avsnitt 4.5.2. Det förutsätts således att huvudformuläret MainForm (4.3.2) redan har tillgång till personobjektet som beskriver aktuell person. Användningsfallet ”Update person” (3.3.4) realiseras (Figur 23) genom att användaren av den personaladministrativa delen av applikationen (3.1.2) väljer att uppdatera aktuell person. MainForm öppnar då aktuell person för editering genom att visa objektet i UpdateUserData (4.3) genom vilken ändringar kan utföras.

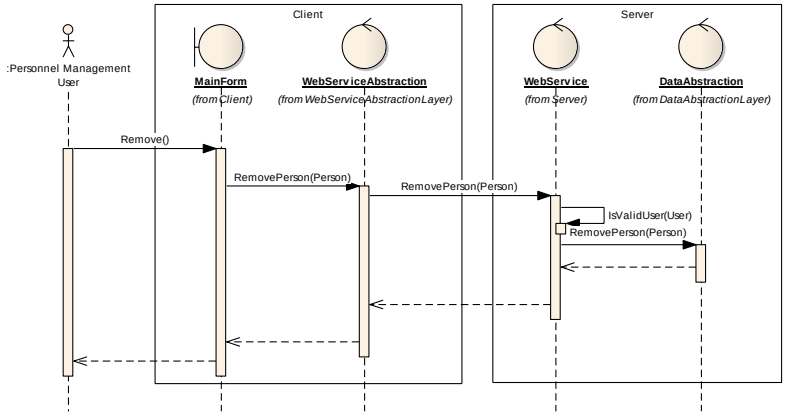



*Figur 23: Realisering av användningsfallet ”Uppdatera person”.*

När användaren är färdig med de ändringar som skulle utföras väljer denne att spara objektet, UpdateUserData anropar då WebServiceAbstraction (4.3.1) och bifogar personobjektet för personen som ska uppdateras. Personobjektet lämnar nu klienten när WebServiceAbstraction i sin tur anropar serverns WebService (4.4.1). WebService kontrollerar att användaren är behörig och om så är fallet så skickas personobjektet vidare till DataAbstraction (4.4.2) för lagring i databasen. Om detta har lyckats hela vägen stängs UppdateUserData och MainForm uppdaterar presentationen av personobjektet.

### 4.5.4 Ta bort person

Att ta bort data om en person görs alltid genom användningsfallet ”Update person” (3.3.4). Att däremot ta bort en person föregås alltid av användningsfallet ”Search for person” (3.3.3) som realiseras i avsnitt 4.5.2. Användningsfallet ”Delete person” (3.3.5) realiseras (Figur 24) genom att användaren för den personaladministrativa delen av applikationen (3.1.2) i MainForm (4.3.2) väljer att ta bort aktuell person.




*Figur 24: Realisering av användningsfallet ”Ta bort person”.*

MainForm anropar då WebServiceAbstraction (4.3.1) och skickar med en unik identifierare för personobjektets som ska tas bort. WebServiceAbstraction anropar serverns WebService som kontrollerar att användaren är behörig och om så är fallet i sin tur anropar DataAbstraction (4.4.2). DataAbstraction rensar all data relaterat till personen och tar slutligen bort all data om personen som nu inte längre existerar i databasen. Om detta går bra hela vägen uppdateras MainForm genom att presentationen av data om personen och själva personobjektet tas bort.

## 4.6 Dataåtkomst

Något som framgår av krav (0) och användningsfall (0) är att alla typer av användare inte ska ha tillgång till samma mängd data. Tabell 1 är resultatet av en analys i syfte att kunna uppfylla det icke-funktionella kravet ”Krav 055 - Gemensam data ska kunna delas av flera delapplikationer med bibehållen sekretess.” (2.2.4) och kan ses som ett förtydligande av tillgång till data utifrån tidigare beskrivna analysklasser. Tabellen visar vilka typer av användare som har tillgång till vilken typ av data. Användarna av den personaladministrativa delen av applikationen kommer som tidigare visats vara de som försörjer samtliga

applikationsdelar med data och är därför de som har den största tillgången till data i applikationen.

| Analysklass | Personnel Management User | Relative User | Phonebook User |
|---|---|---|---|
| Person | ✓ | ! | ! |
| Address | ✓ | ✓ | ! |
| PhoneNumber | ✓ | ✓ | ! |
| EmailAddress | ✓ | ✓ | ! |
| OrganisationUnit | ✓ | ✓ | ! |
| Relation | ✓ | ✓ | ✗ |
| IdentificationCard | ✓ | ✗ | ✗ |
| Passport | ✓ | ✗ | ✗ |
| RegisterControl | ✓ | ✗ | ✗ |
| DrivingLicense | ✓ | ✗ | ✗ |
| MilitaryDrivingLicense | ✓ | ✗ | ✗ |
| Education | ✓ | ✗ | ✗ |
| EducationalInstitution | ✓ | ✗ | ✗ |
| ExerciseParticipation | ✓ | ✗ | ✗ |
| Exercise | ✓ | ✗ | ✗ |
| MissionParticipation | ✓ | ✗ | ✗ |
| Mission | ✓ | ✗ | ✗ |
| BankAccount | ✓ | ✗ | ✗ |

✓ Full tillgång till all data
! Begränsad tillgång
✗ Ingen tillgång

*Tabell 1: Olika användartypers åtkomst till data.*

Efter de personaladministrativa användarna kommer användarna av anhörigdelen av applikationen som har tillgång till nästan alla personklassens attribut och de data i de kollektioner som innehåller adresser, telefonnummer och e-postadresser. Dessa användare har även tillgång till data om i applikationen angivna relationer och tillåts därigenom att nå motsvarande data, beskriven ovan, även om respektive anhörigperson. Det som gör att denna användartyp inte har full tillgång till Person är att de inte har tillgång till personens personnummer.

Lägst behörighet har användarna av telefonkatalogen, som sannolikt till antal utgör den största gruppen, de har bara rätt att komma åt delar av personklassens attribut och data från de

kollektioner som innehåller arbetsrelaterade adresser, telefonnummer och e-postadresser, det vill säga de delar som användarna av den personaladministrativa delen av applikationen har valt att visa.

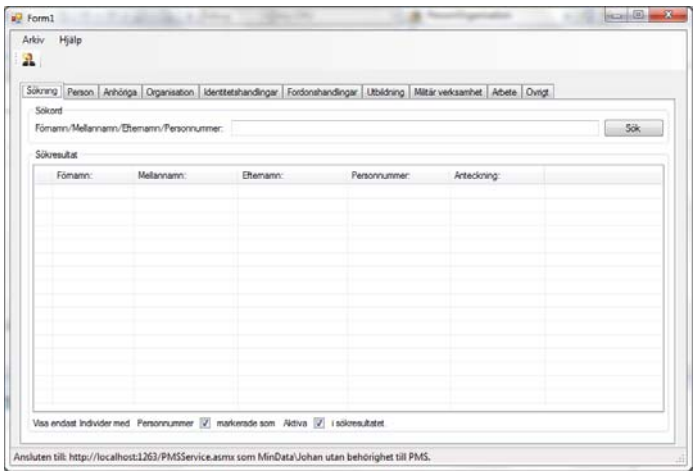
## 4.7 Användargränssnitt

Efter framställning av en enkel användningsfallsmodell började gränssnitten skisseras, först med papper och penna och sedan direkt i Microsoft Visual Studio. När skisserna började ta form avhandlades de efterhand med kundens representant som gav sina synpunkter och i vissa fall hämtade in ytterligare synpunkter så att gränssnitten därefter kunde utvecklas och förfinas för att på så sätt kunna uppfylla de ickefunktionella kraven på användbarhet (2.2.5).
Fokus har vid framtagandet varit att göra gränssnitten tydliga, enkla och intuitiva för att de på så sätt ska kännas användarvänliga något som dessutom förkortar den utbildningstid som krävs för att lära sig använda applikationen för att på så sätt kunna uppnå de krav som ställts på användbarhet (2.2.5). För att säkerställa att så är fallet så har gränssnitten löpande provats av några av de tänkta användarna av applikationen.

Utifrån givna prioriteringar (1.5) så är de gränssnitt som presenteras i följande avsnitt endast knutna till klienten för den personaladministrativa delen av applikationen och till analysklassen MainForm (4.3.2) som beskriver huvudfönstret för applikationen.

### 4.7.1 Användargränssnitt för sökning av person

Gränssnittet för sökning (Figur 25) är den del av den personaladministrativa delen av applikationen i vilken önskad person söks ut. Sökning kan ske genom att skriva in del av eller hela personens förnamn, mellannamn, eller personnummer.



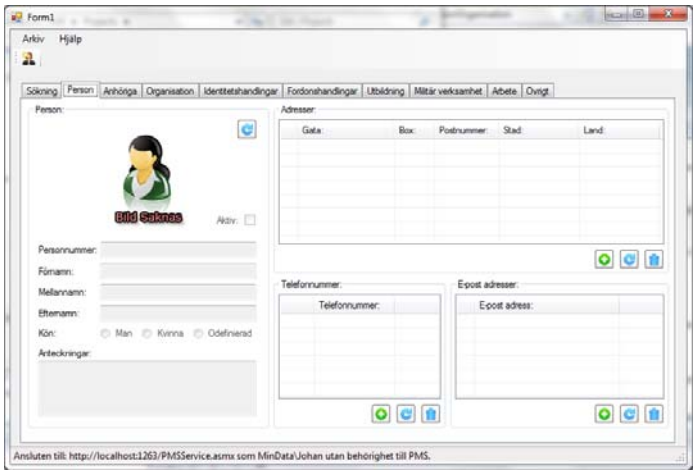

*Figur 25: Gränssnitt för sökning av person.*

Personen väljs sedan ur den lista som utgör sökresultatet. Vald person hämtas då från databasen. De filter som kan anges längst ner i fönstret är till för att kunna välja att endast se anställda då de är de enda som lagrats med personnummer i databasen alternativt de som är aktiva som utgörs av just nu tjänstgörande personal. För ökad spårbarhet är tanken att de som inte längre är kopplade till verksamheten inte ska tas bort ur systemet utan endast inaktiveras. Anhöriga kan också sökas direkt i detta gränssnitt. Anhöriga kan naturligtvis också vara anställda. När önskad person hittats väljs denne genom dubbelklick med musen. Detta val resulterar i att användargränssnittet för hantering av Person synliggörs ifyllt med data för vald person (Figur 26).

### 4.7.2 Användargränssnitt för hantering av person

Via gränssnittet för hantering av person (Figur 26) möjliggörs hantering av data om en person, dennes adress, telefonnummer och e-postadresser.

I den vänstra delen av gränssnittet visas data som hämtas ur attribut i objektet person (4.2.1) dessa kan uppdateras genom att knappen för att uppdatera en person (pilen) klickas. Till höger listas adressobjekt (4.2.2), telefonnummerobjekt (4.2.3) och e-postadressobjekt (4.2.4) ur respektive kollektioner från objektet person. Det går att lägga till, uppdatera och ta bort ytterligare adresser, telefonnummer, och e-postadresser genom att klicka på de knappar som finns under respektive presentation. Knappen med plustecknet är för att lägga till, knappen med pilen för att uppdatera och knappen med soptunnan för att ta bort.

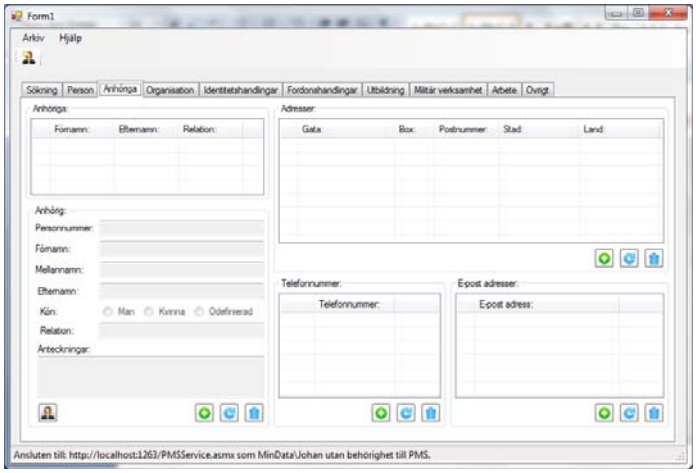



*Figur 26: Personflik i den personaladministrativa delen av applikationen.*

I det inledande arbetet med gränssnittet kunde en person vara av odefinierat kön men då så sällan är fallet i verkligheten valdes därför under implementation att endast behålla man och kvinna där defaultvärdet är kvinna.

### 4.7.3 Användargränssnitt för hantering av anhöriga

Användargränssnittet för hantering av anhöriga (Figur 27) liknar mycket det för hantering av person med några få skillnader, i stället för bilden som finns i persongränssnittet så finns en lista på anhöriga till den valda personen. Ur listan väljs sedan den anhörige vars data man önskar förändra. En skillnad från användargränssnittet för hantering av person är att de knappar under presentationen av data om den anhörige är fler än för person. Dessa knappar är till för att lägga till en ny anhörig (plustecknet), uppdatera anhörig (pilen) eller ta bort anhörig (soptunnan). Ta bort innebär inte att den anhörige tas bort utan endast att relationen som finns mellan de två personerna tas bort. Själva personen som den anhörige utgör tas bort på samma sätt som alla andra personer i applikationen tas bort, via gränssnittet för hantering av Person. Den sista skillnaden är den knapp med en person på som öppnar ett sökfönster och därmed möjliggör sökning bland befintliga personer i applikationen för att på detta sätt kunna skapa en ny relation (4.2.8) mellan två individer.




*Figur 27: Anhörigflik i den personaladministrativa delen av applikationen.*

Den högra sidan av gränssnittet visar adresser, telefonnummer och e-postadresser för vald anhörig och är identisk med den högra sidan av personfliken (Figur 26) men är i detta fall för den anhörige.

# 5 Design och implementation

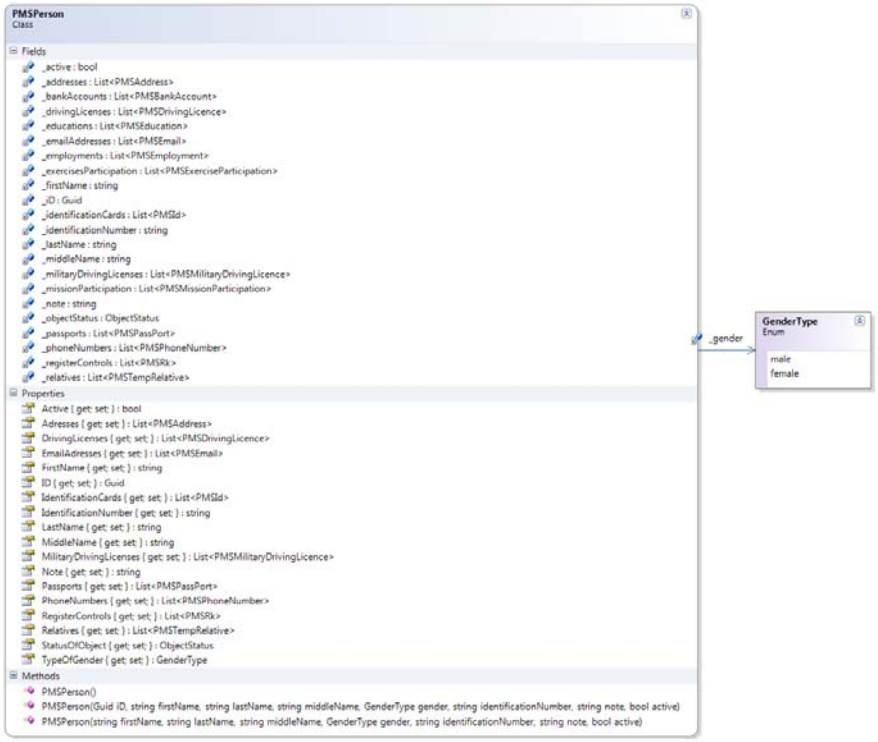
Med utgångspunkt i analysen och påverkande icke-funktionella krav så används en gemensam databas för samtliga applikationsdelar som skapas i Microsoft SQL Server 2005. Klientdelen för den personadministrativa delen kommer som framgår av analysen (4.1) att implementeras som en Microsoft Windows Forms-applikation medan anhörigdelen och telefonkatalogens klienter är tänkta att implementeras webbaserade med Microsoft ASP.NET och köras på en Internet Information Server. De olika applikationsdelarna kommunicerar sedan med databasen via en gemensam webbtjänst. Webbtjänsten innehåller förutom databaskommunicerande delar också metoder för behörighetskontroll och dataåtkomst. All design och implementation förutom skapandet av databasen genomförs i utvecklingsverktyget Microsoft Visual Studio 2008 Professional och all kod skrivs i det objektorienterade programspråket C#.

## 5.1 Gemensamma klasser

En grundförutsättning för hela applikationen är som tidigare nämnts hur ett personobjekt sätts samman och de klasser som används för att beskriva en person kommer att användas både av servern och av klienterna för de olika applikationsdelarna. De har därför samlats i ett eget programbibliotek som kallas PMSSharedObjects. I resten av detta avsnitt beskrivs några av de klasser som ingår i PMSSharedObjekts.

### 5.1.1 Klassen PMSPerson

Som framgick redan vid analysen så är personklassen en viktig del då den håller samman all data om en person. Implementationen av PMSPerson är baserad på analysklassen Person (4.2.1) och beskrivs av klassdiagrammet i Figur 28.



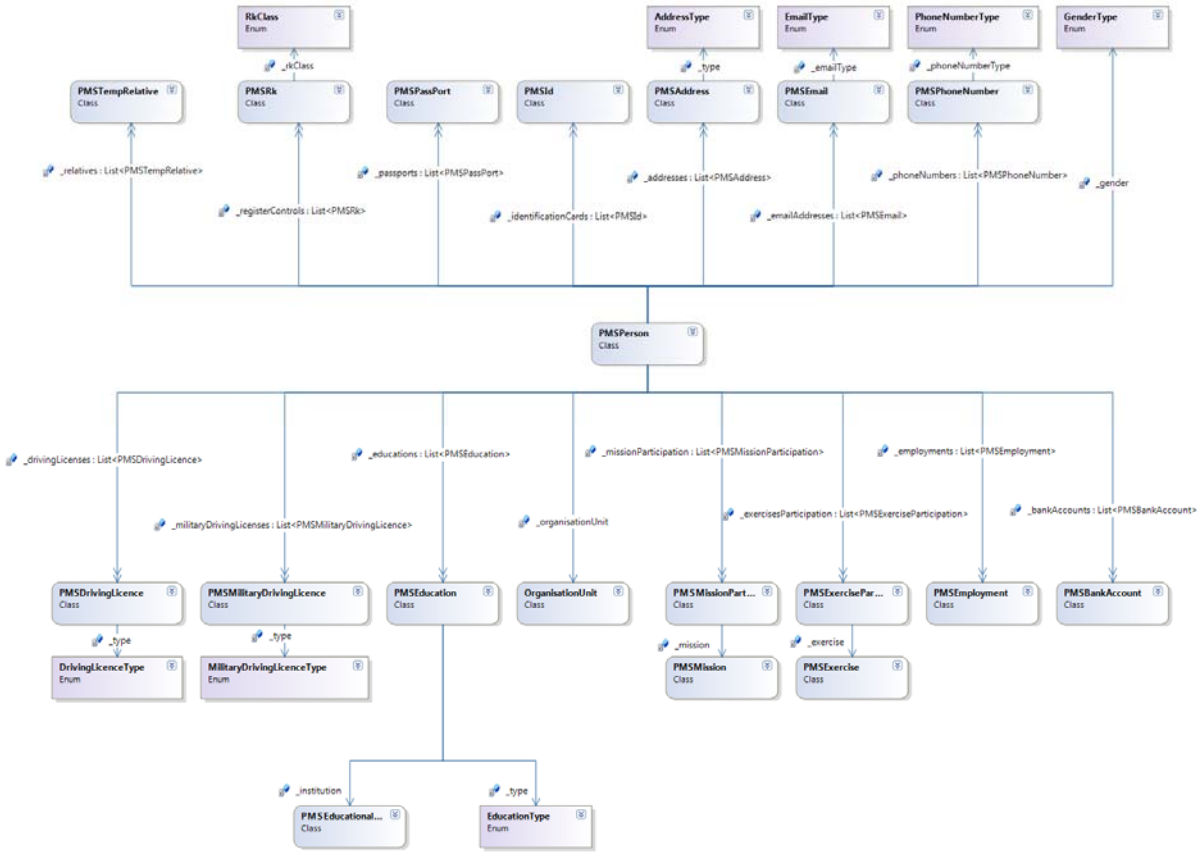

*Figur 28: Klassen PMSPerson.*

Denna bild visar dock inte den kompletta klassen. Klassen PMSPerson används för att hantera data både om personer och anhöriga.

Bland de data som PMSPerson hanterar återfinns de grundläggande personattributen. Där finns förnamn, efternamn, mellannamn och personnummer som är av typen string. Personen måste ha en unik identifierare för att kunna särskilja den från andra personer, denna identifierare är av typen Globally unique identifier (Guid) och lagras i variabeln _iD. En Guid representeras som ett 32-tecken långt hexadecimalt tal, som vanligtvis lagras som en 128-bitars integer. Talet blir så stort att sannolikheten att samma tal slumpmässigt genereras två gånger är försumbar [16]. För att beskriva personens kön används typen GenderType som är en enum som kan anta värdet male eller female. Personens kön lagras i variabeln _gender. Till de grundläggande personattributen har också tillförts möjligheten att göra en anteckning som är av typen string och lagras i variabeln _note.

För att möjliggöra aktivering och inaktivering av en person finns variabeln _active som utgörs av en boolean, en datatyp som kan anta värdet sant (true) eller falskt (false).

Hur personobjektet sätts samman skiljer lite mellan de olika applikationsdelarna men för den personaladministrativa delen av applikationen som innehåller alla delar så ser klassdiagrammet ut som nedan (Figur 29).




*Figur 29: Klassdiagram för PMSPerson*

De klasser av vilka det kan finnas fler av än en hanteras, som redan beskrivits i analysen (4.2.1), i klassen PMSPerson som kollektioner. Dessa kollektioner utgörs av typade listor. Dessa listor kan bara innehålla en specifik typ av objekt som fastställs när listan skapas. Kollektioner visas i Figur 29 som pilar med dubbla pilspetsar. Exempel på dessa kollektioner är listan för att hantera adresser som i klassen PMSPerson (Figur 28) visas som List<PMSAddresss>.

### 5.1.2 Klassen PMSTempPerson

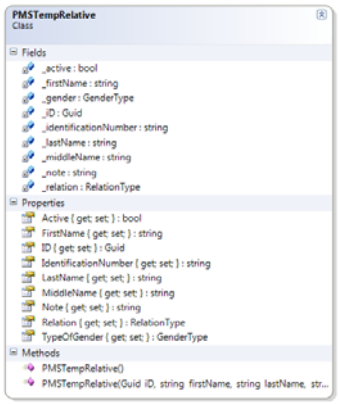
Klassen PMSTempPerson (Figur 30) har ingen motsvarande analysklass. Klassen används för att hantera en Lista av temporära personer som är resultatet av en genomförd sökning efter personer i databasen.


*Figur 30: Klassen PMSTempPerson.*

Klassen används bara av den personaladministrativa delen av applikationen och innehåller bara de data som utgör de grundläggande personattributen.

### 5.1.3 Klassen PMSTempRelative

Klassen PMSTempRelative (Figur 31) har inte heller någon motsvarande analysklass och är snarlik klassen PMSTempPerson (5.1.2) med ett undantag, den innehåller också en relation (5.1.4). Klassen används i den lista som representerar en persons anhöriga i klienten.

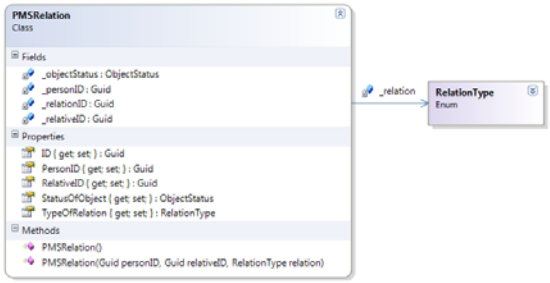



*Figur 31: Klassen PMSTempRelative.*

Den är bara till för presentation så vid behov av mer data om den anhörige så hämtas ett fullständigt personobjekt, en instans av PMSPerson (5.1.1), från databasen.

### 5.1.4 Klassen PMSRelation

Klassen PMSRelation (Figur 33) baseras på analysklassen Relation (4.2.8) och beskriver en relation mellan två personer i applikationen.

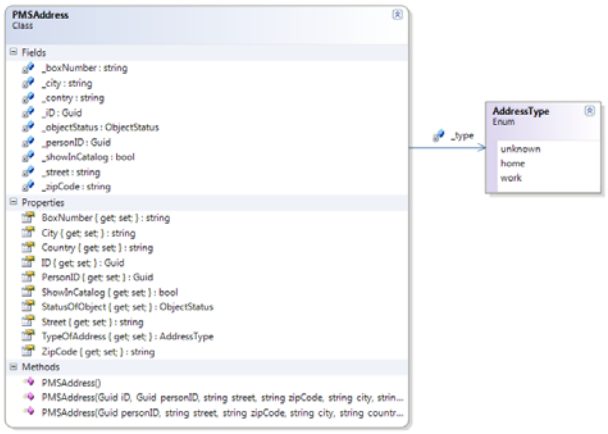



*Figur 32: Klassdiagram för PMSRelation*

Utgångspunkten är den aktuella person vars unika identifierare benämns _personID, denne har en relation med personen vars unika identifierare benämns _relativeID. Relationen beskrivs av variabeln _relation som utgörs av en Enum benämnd RelationType. RelationType beskriver ett antal fördefinierade relationer som bland annat mor, far, bror, syster, make, maka, pojkvän och flickvän. Relationen är enkelriktad och variabeln _relation beskriver vilken roll personen som är anhörig har till den aktuella personen.

### 5.1.5 Klassen PMSAddress

Klassen PMSAddress (Figur 33) är baserad på analysklassen Address (4.2.2) den innehåller attribut för att beskriva en adress och metoder för att ändra dessa. Varje adress är knuten till en specifik person vars unika identifierare lagras i variabeln _personID. En annan Guid innehåller adressens egen unika identifierare och lagras i variabeln _iD.



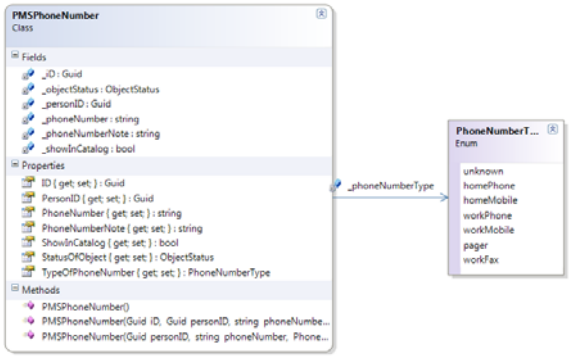

*Figur 33: Klassdiagram för PMSAddress*

PMSAddress är förutom PMSPerson en av de klasser som kommer att användas av samtliga delar av applikationen. I anhörigdelen av applikationen kommer samtliga adresser att visas för vald person och för dennes anhöriga. I telefonkatalogen styrs presentationen av huruvida adressen ska visas eller inte av variabeln _showInCatalog. Variabeln är av typen boolean och

kan därmed anta värdet sant (true) eller falskt (false). Även om variablen skulle vara sann visas adressen bara under förutsättning att den är arbetsrelaterad . Detta styrs av variabeln _type, som är av typen AddressType och som för att visa adressen ska vara AddressType.work .

### 5.1.6 Klassen PMSPhoneNumber

Klassen PMSPhoneNumber är baserad på analysklassen PhoneNumber (4.2.3) och beskriver ett telefonnummer. PMSPhoneNumber är en av de klasser som kommer att användas av samtliga delar av applikationen.

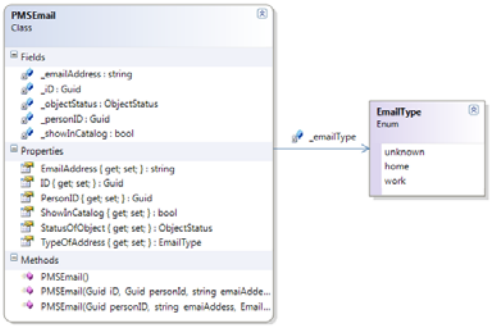



*Figur 34: Klassdiagram för PMSPhoneNumber*

I anhörigdelen av applikationen kommer samtliga telefonnummer att visas för vald person och för dennes anhöriga, precis som för adresser (5.1.5). I telefonkatalogen styrs presentationen av huruvida telefonnumret ska visas eller inte av variabeln _showInCatalog men precis som för adresser (5.1.5) krävs att numret är arbetsrelaterat. Detta styrs av variabeln _phoneNumberType, som är av typen PhoneNumberType och ska ha värdet workPhone, workMobile, pager eller workFax för att numret ska visas .

### 5.1.7 Klassen PMSEmail

Klassen PMSEmail är baserad på analysklassen EmailAddress (4.2.4) och beskriver en e-postadress. Den är också en av de klasser som används av samtliga delar av applikationen och uppvisar stora likheter med PMSAddress (5.1.5) och PMSPhoneNumber (5.1.6).

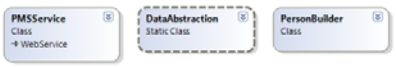



*Figur 35: Klassdiagram för PMSEmail*

I anhörigdelen av applikationen kommer samtliga e-postadresser att visas för vald person och för dennes anhöriga, precis som för adresser (5.1.5) och telefonnummer (5.1.6) och på samma sätt som i dessa klasser så styrs presentationen av huruvida e-postadressen ska visas eller inte i telefonkatalogen av variabeln _showInCatalog men bara under förutsättning att e-postadressen är arbetsrelaterad.

## 5.2 Server

Serverdelen för den personaladministrativa delen av applikationen har utökats från hur den såg ut efter analysen. Den består nu av tre delar (Figur 36) i stället för två. De delar som ingår är webbtjänsten, PMSService (5.2.1), som medger åtkomst till de metoder som klienten använder för att kommunicera med servern vilket sker under förutsättning att användaren är behörig. Sedan följer DataAbstraction (5.2.2) som är den del som hanterar transformationen mellan databasens data och applikationens objekt. Den del som tillkommit är resultatet av att klassen DataAbstraction har delats upp i två klasser där DataAbstraction svarar för all hantering mot databasen utom hämtning och sammansättning av data för att skapa fullständiga personobjekt detta sköts istället av klassen PersonBuilder (5.2.3).

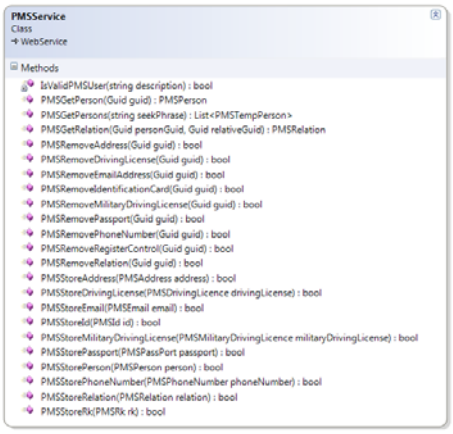



*Figur 36: Klassdiagram för servern.*

### 5.2.1 Klassen PMSService

PMSService (Figur 37) är den klass som utgör webbservicen och är baserad på analysklassen WebService (4.4.1). Klassen utgör klientens gränssnitt mot servern och innehåller alla webbmetoder som klienten behöver bland annat för att lägga till personer, söka efter personer, uppdatera personer och ta bort personer. Den enda metod i klassen som inte är en webbmetod

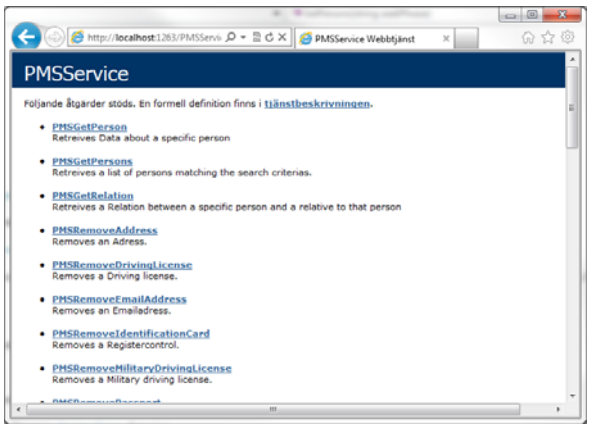
är IsValidPMSUser. Denna metod anropas alltid först i alla webbmetoder för att kontrollera att anropet sker från en behörig användare och om så inte skulle vara fallet så kommer inte webbmetoden att köras.




*Figur 37: Klassen PMSService.*

Webbmetoderna i PMSService har ingen annan funktion än att anropa IsValidPMSUser för att kontrollera användarens behörighet och när detta är gjort och om användaren är behörig så anropas en metod med motsvarande namn i klassen DataAbstraction (4.4.2). Detta innebär att alla metoder i PMSService utom IsValidPMSUser har en korresponderande metod i databasabstraktionsklassen. Klassdiagrammet i Figur 37 visar inte alla i klassen ingående metoder.

Webbtjänsten som PMSService utgör är åtkomlig direkt via webbläsare (Figur 38) genom vilken också vissa av webbmetoderna kan provköras direkt beroende på vilken typ av indata de kräver.

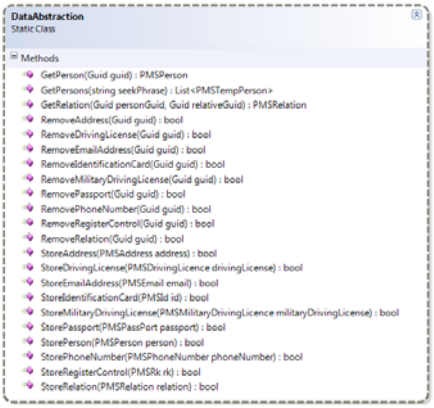



*Figur 38: PMSService via webbläsare.*

Det går också att genom denna webbsida se de XML-formaterade Simple Object Access Protocol (SOAP) meddelanden [17] som utgör kommunikationen till och från webbtjänsten.

### 5.2.2 Klassen DataAbstraction

DataAbstraction (Figur 39) är baserad på analysklassen med samma namn (4.4.2). Klassen är implementerad som static vilket innebär att en instans av den skapas vid serverns start och är tillgänglig under hela tiden som servern körs och dess metoder kan anropas utan att ett objekt behöver skapas för att de ska vara åtkomliga.

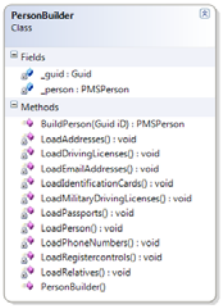



*Figur 39: Klassen DataAbstraction.*

Klassen innehåller metoder för att spara objekt till databasen och för att sätta samman data i databasen till objekt. Metoderna i klassen använder inte SQL för detta utan använder i stället ADO.NET Entity Framework [18] i kombination med LINQ [19]. Klassdiagrammet i Figur 39 visar ett urval av de metoder som ingår.

### 5.2.3 PersonBuilder

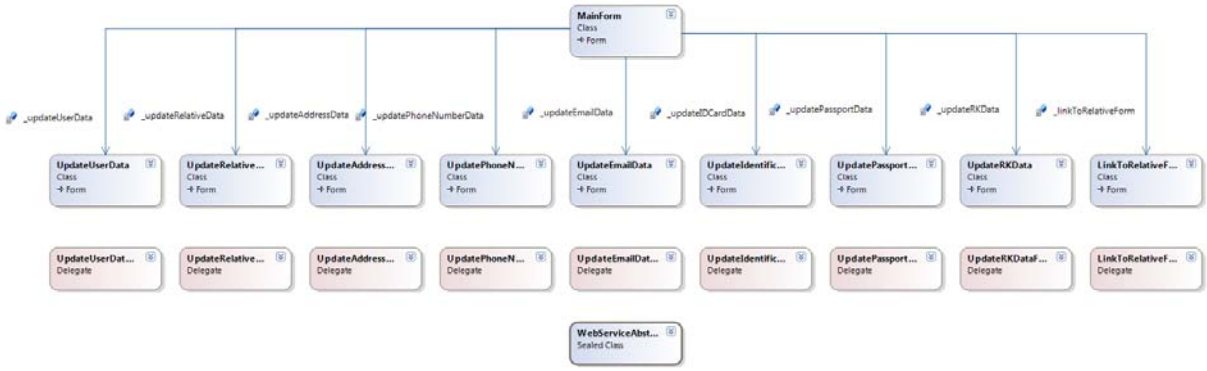
Klassen PersonBuilder (Figur 40) tillkom för att bryta ut de delar av DataAbstraction som sätter samman ett personobjekt utifrån det data som är lagrat i databasen om en specifik person. Detta gjordes ursprungligen i bara en metod som blev så stor att den upplevdes oöverskådlig. Den bröts därför ned i flera nya metoder som medförde att koden blev lättare att följa. Klassdiagrammet i Figur 40 visar inte alla i klassen ingående metoder.




*Figur 40: Klassen PersonBuilder.*

## 5.3 Klient

Klienten (Figur 41) omfattar ett antal olika Windows Forms som utgör gränssnitten mot användaren för att presentera och manipulera de olika gemensamma objekt av vilka klasserna för några beskrivits i avsnitt 5.1. Klienten består också av ett abstraktionslager för att hantera transformering av data mot webbtjänsten (5.2.1).




*Figur 41: Klassdiagram för klient*

I resterande del av detta avsnitt så kommer några av gränssnitten (5.3.1) och klassen WebServiceAbstraction (5.3.2) att presenteras närmare

### 5.3.1 Windows forms

Fokus vid design av gränssnitt har som tidigare beskrivits vara att få gränssnitten att kännas enkla och intuitiva (4.7). Nästan alla av de klasser som används för att skapa datarepresentationen för en person behöver ett eget gränssnitt för att data ska kunna läggas till, uppdateras eller tas bort.

För att hantera data för objekt skapade från klassen PMSPerson (5.1.1) används klassen UpdateUserData (Figur 42), ett Windows Form, som utgör gränssnittet för detta.


*Figur 42: Gränssnitt för UpdateUserData*

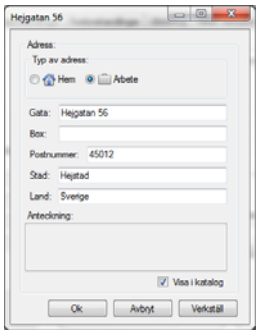
I gränssnittet finns fält som motsvarar de grundläggande personattributen. Övriga data som till exempel då adresser, telefonnummer och e-postadresser hanteras genom andra gränssnitt (Figur 41). För att skriva persondata till databasen anropas en metod i WebServiceAbstraction (5.3.2) som heter StorePerson till vilken hela personobjektet skickas med (Figur 43).

```
{
    success = WebServiceAbstractionLayer.WebServiceAbstraction.Instance.StorePerson(_person);

    if (success)
```

*Figur 43: Anrop till StorePerson i WebServiceAdstraction*

Eftersom möjlighet att hantera data för adresser inte infogats i UpdateUserData så har ett annat gränssnitt för att göra detta tagits fram, det är en Windows Form klass som heter just UpdateAddressData (Figur 44).




*Figur 44: Gränssnitt för UpdateAddressData.*

Gränssnittets fält i denna klass för att lägga till, uppdatera och ta bort data på adressobjektet överensstämmer på samma sätt som för PMSPerson med de metoder som finns i klassen PMSAddress (5.1.5). För att skriva adressdata till databasen anropas en metod i

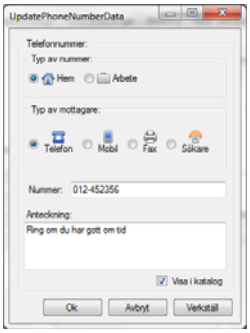
WebServiceAbsraction som heter StoreAddress till vilken ett adressobjekt skickas med (Figur 45).

```
if (_address.StatusOfObject != PMSSharedObjects.Meta.ObjectStatus.unchanged)
{
    success = WebServiceAbstractionLayer.WebServiceAbstraction.Instance.StoreAddress(_address);
    if (success)
```

*Figur 45: Anrop till StoreAddress i WebServiceAdstraction*

Det är alltså bara adressobjektet som kommuniceras till servern och skapas eller uppdateras i databasen, inte hela personobjektet

Gränssnitten för att hantera telefonnummer (Figur 46) och e-postadresser (Figur 48) är uppbyggda på samma sätt som UpdateAddress. De har också precis som UpdateAddressData motsvarande metoder de anropar i WebServiceAbstraction.




*Figur 46: Gränssnitt för UpdatePhoneNumberData.*

När ett nyskapat telefonnummer för en specifik person ska sparas så anropar klassen UpdatePhoneNumberData metoden StorePhoneNumber i WebServiceAbstraction och telefonnummerobjektet bifogas (Figur 47).

```
if (_phoneNumber.StatusOfObject != PMSSharedObjects.Meta.ObjectStatus.unchanged)
{
    success = WebServiceAbstractionLayer.WebServiceAbstraction.Instance.StorePhoneNumber(_phoneNumber);
    if (success)
```

*Figur 47: Anrop till StorePhoneNumber i WebServiceAdstraction*

Det samma gäller för klassen UpdateEmailData som anropar motsvarande metod StoreEmail i WebServiceAbstraction.

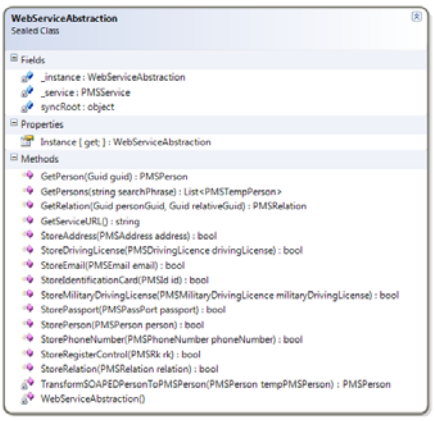

*Figur 48: Gränssnitt för UpdateEmailData.*

Gemensamt för de Windows Forms som presenterats är att de öppnas via MainForm, och att de själva anropar instansen av WebServiceAbstraction för att spara de objekt som skapats eller uppdateras. De implementerar också en delegat [20] som möjliggör för MainForm att veta när de stängs så att huvudgränssnittet kan uppdateras. Gemensamt är också att de har valet om objektet ska visas i telefonkatalogen eller inte.

Gemensamt för samtliga Windows Forms som används för att representera olika typer av dataobjekt är att de använder metoder i WebServiceAbstraction för att spara den specifika typ av objekt som de hanterar utan att skicka med något personobjekt. Detta möjliggörs genom att den unika identifieraren för den specifika person som till exempel adressen tillhör också återfinns i adressobjektet.

### 5.3.2 WebServiceAbstraction

All kommunikation mot webbtjänsten sköts via klassen WebServiceAbstraction (Figur 49). Klassen är implementerad enligt designmönstret Singleton vilket innebär att det endast finns en instans av klassen i applikationen. Instansen skapas vid första anropet på någon av dess metoder och vid ytterligare anrop returneras bara referensen till den initialt skapade instansen [21]. Klassen har som enda uppgift att utgöra det kommunicerande gränssnittet mot serverns webbtjänst. Den innehåller metoder för att spara nästan varje enskild objekttyp som återfinns i biblioteket PMSSharedObjects som delas av både server och klient av vilka några redovisats i avsnittet ovan (Figur 43, Figur 45 och Figur 47).

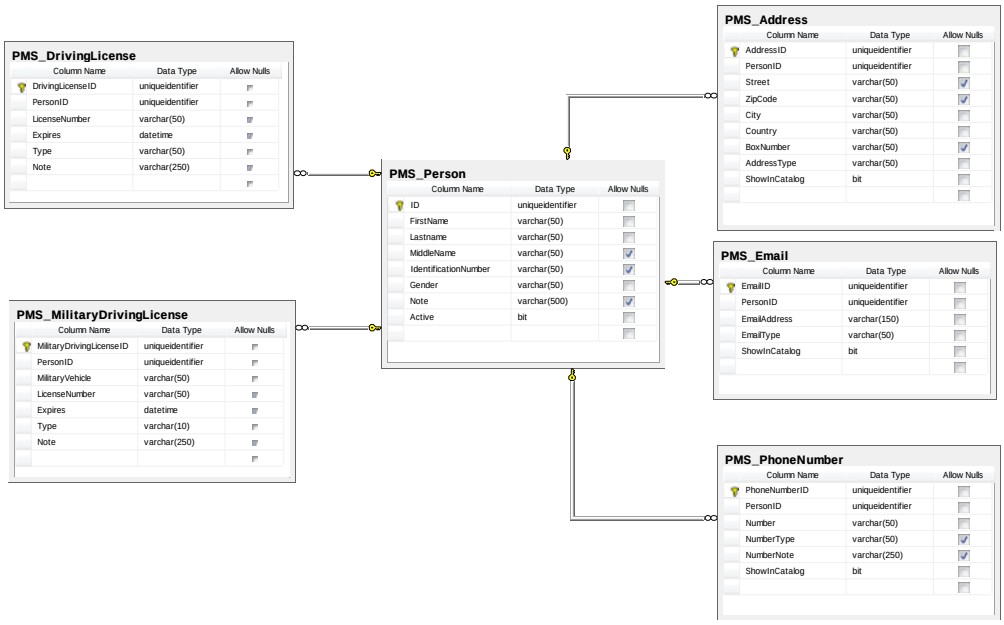



*Figur 49: Klassen WebServiceAbstraction*

## 5.4 Databas

Databasen för att lagra data om personer byggdes efter det att stommen till klasserna hade skapats och innan valet att använda ADO.NET Entity Framework hade gjorts. Detta har resulterat i en del omvandlingar som kan kännas onödiga mellan de objekt som har skapats av ADO.NET Entity Framework och de objekt som skapats i PMSSharedObjects. Detta hade kanske kunnat undvikas om databasen hade skapats först men det hade också krävt att ADO.NET Entity Framework varit känt när arbetet påbörjades vilket det inte var.

För att exemplifiera databasens enkla uppbyggnad presenteras i Figur 50 några av de tabeller som används för att lagra data om de objekt vars klasser beskrivs i avsnitt 5.1. I figuren återfinns tabeller som PMS_Person som motsvarar klassen PMSPerson, PMS_Address som motsvaras av klassen PMSAddress och så vidare för samtliga klasser som objekt ska lagras ifrån.




*Figur 50: Några tabeller i databasen*

## 5.5 Slutsatser relaterade till design och implementation

Vad gäller både design och implementation av applikationen så har det kunnat genomföras relativt problemfritt. Det var endast två delar av lösningen som var nya för författaren och det var användning av ADO.NET Entity Framework och av webbtjänster.

Vid en uppdatering av applikationen är det dock några saker som kan behöva förändras, till exempel så bör alla komponenter av typen ListView bytas ut mot DataGridViews då dessa kan kopplas direkt till de typer av listor som används i applikationen. Det blir då möjligt att enklare styra vilka fält i en klass som ska visas eller inte. Detta har nu lösts med att till exempel kolumnlängden för kolumnen som visar objektens unika identifierare i ListView:n är satt till 0 då den måste finnas med men inte ska synas för användaren. Objektens unika identifierare är den enda referens som kan antas vara unik och den används därför för att hitta det objekt som ett visst ListViewItem refererar till. I en DataGridView refereras istället direkt till det aktuella objektet.

Det bör också tillföras kontroller för att validera det data som matas in i de olika gränssnitten. Vid inmatning kontrolleras som applikationen nu är utformad inte att det till exempel är ett giltigt telefonnummer eller en giltig e-postadress som matats in utan denna kontroll är helt

upp till användaren. Dessa typer av kontroller är dock något som bör tillföras både för att underlätta för användaren och för att förhindra eventuella fel som kan uppstå i applikationen.

Applikationen hanterar som den nu är utformad inte heller metadata om till exempel händelser så som när person lades in i applikationen, när den senast uppdaterades och så vidare. Ur ett administrativt perspektiv så skulle denna typ av data kunna vara värdefull då systemet varit i drift en längre period och man behöver veta om data föråldrats.

Något som kanske inte framgår så tydligt genom detta arbete är att anställningsnumret hanteras utanför själva personobjektet både i objektmodellen och i databasmodellen, en sannolik tankevurpa, men synen på hur detta borde se ut i applikationen har förändrats under arbetets fortskridande och en ändring bör genomföras. En sådan förändring skulle bland annat underlätta den något otydliga filtreringen av personer vid sökning som framgår i Figur 25 där filtrering kan göras på personnummer då detta initialt ansågs vara det som skilde anställda från anhöriga. Förändringen som ännu inte genomförts kommer att påverka klasserna PMSPerson (5.1.1), PMSTempPerson (5.1.2) och databastabellen PMS_Person (5.4), så tillvida att ett attribut för att lagra ett anställningsnummer kommer att tillföras. Det kommer också att påverka en databastabell vid namn PMS_PersonExtra ur vilken det nu befintliga attributet kommer att tas bort.

# 6 Slutsatser

Utvecklingen av Personnel Management System, PMS, kommer av Försvarsmaktens telenät och markteleförbands behov att ersätta en befintlig applikation skapad i Microsoft Access. Man hade ett behov av att hantera personal, så som frivilliganställda och tillsyningsmän, som inte ryms i det vanliga systemet för personalhantering. Tanken var att applikationen förutom att kunna hantera nämnda personal också skulle kunna användas av vakthavande befäl för att vid behov kunna söka efter en persons anhöriga och som en telefonkatalog.

Fokus under denna utvecklingsprocess har varit att försöka tillgodose användarnas uttalade behov genom att skapa ett system för att hantera personal. Syftet som varit tydligt uttalat har tillsammans med beställarens representant kunnat brytas ner i de krav på funktioner som beskrivs av de funktionella kraven i kapitel 0 och som sedan förtydligas med hjälp av användningsfallen i kapitel 0. Genom analysen i kapitel 0 har en arkitektur skapats för att kunna realisera användningsfall och ta hand om de icke-funktionella krav som också beskrivs i kapitel 0. I kapitel 0 har designen ytterligare detaljerat den genomförda analysen och implementation har genomförts. Den uppmärksamme undrar nu om testning genomförts vilket det naturligtvis har, däremot så har ingen särskild testfallsspecifikation tagits fram. Alla tester har genomförts på modulnivå för att säkerställa modulernas funktion. Det bör dock påpekas att verifiering av applikationen ännu inte genomförts när detta arbete slutförts och att det inte innebär något större problem att generera en testfallsspecifikation utifrån beskrivna användningsfall

En av svårigheterna med att skriva kraven var att inte frestas att kravställa den systemlösning eller den applikation man själv avser utveckla. Det viktigaste med kraven är istället att de fångar kravställarens behov. Dialogen runt behoven kan mycket väl röra det aktuella systemet men när kraven skrivs är det viktigt att perspektivet lyfts tillbaka till behovet och att det är detta som kravställs så att det inte i kravställningen bakas in något designmässigt fel från den tänkta systemlösningen som diskuterats och som senare vid upptäckt inte enkelt kan kringgås för att felaktigheten är kravställd i kravspecifikationen. Detta är dessutom något som tydliggörs om man betraktar Berndt Brehmers teori om designlogik, i vilken han menar att ledningssystem kan analyseras i tre nivåer där den högsta nivån utgörs av ”Syfte” som svarar på frågan ”varför?” det vill säga varför systemet existerar eller varför ska det utvecklas. Nästa nivå utgörs av ”Funktion” och som svarar på frågan ”vad?”, det vill säga vad som måste göras

för att uppnå syftet. Den sista nivån som utgörs av ”Form” som svarar på frågan ”hur?” det vill säga hur systemet uppfyller funktionerna så att syftet kan uppnås [22]. Kraven bör således hållas på funktionsnivå så långt det är möjligt snarare än på formnivå, något som inte alltid är möjligt så därav icke-funktionella krav. Den viktigaste slutsatsen är dock att utan ett tydligt uttalat syfte är det svårt att skapa funktionella krav och ju mer avgränsat ett syfte är desto lättare blir det att definiera funktioner för att det ska uppnås. Det är med andra ord i princip omöjligt att skapa ett omnipotent system.

Genom att fokus varit att tillgodose användarnas behov så har också tyngdpunkten i detta arbete förskjutits från själva implementationen till krav, användningsfall och analys. Avsikten har därför inte varit att skapa någon optimal kod utan att istället med hjälp av modeller skapa en fungerande dialog med användaren för att närmare förstå behoven. Det har redan från början kalkylerats med att det måste till flera iterationer för att färdigställa applikationen där både förändringar av krav och förändringar av problemlösning relaterat till applikationen måste kunna förekomma och utvecklingen av applikationen slutar inte i samband med detta arbetes slutförande.

En farhåga, då verksamhetens målbild för applikationen antas utvecklas parallellt med utvecklingen av applikationen, innebär att kraven kommer att utgöra ett slags snapshot för hur verksamheten upplevde sina behov under den tidsperiod då dessa genererades. Det innebär också att verksamheten aldrig kommer att uppleva att systemet är ”rätt system” vid en validering då systemet alltid ligger efter i förhållande till den egna målbilden. Verifiering, som görs mot kraven om dessa är tillfredställande hanterade, kommer däremot sannolikt inte att utgöra några problem. För att ovan beskrivna scenario inte ska inträffa krävs en kort utvecklingsprocess vilket gör att förändringen av den initiala målbilden inte hinner bli lika stor. För projekt som sträcker sig över längre tidsperioder kan lösningen vara att utveckling av flera versioner av samma system pågår parallellt där varje version utgår från ett eget snapshot med krav, ett förfarande som sannolikt bör pågå under hela systemets livscykel, fram till dess systemet avvecklas för att därmed hindra systemet från att stagnera. Den kreativa processen kan då generera krav till nästa version snarare än att den påverkar och förändrar krav i den version som är under framtagande.

I den framtida utvecklingen av applikationen ligger närmast ett färdigställande av fullständig funktionalitet enligt steg 1, det vill säga grundläggande funktionalitet i den

personaladministrativa delen av applikationen, därefter installation i testmiljö för verifiering och validering. Applikationen ska också tillföras steg 2 och steg 3, som omfattade färdigställande av anhörigdelen av applikationen, telefonkatalogen och en komplettering för hantering av ytterligare data i den personaladministrativa delen. Applikationen kommer med stor sannolikhet också att genomgå flera ytterligare iterationer med inarbetning av tillförda krav under flera år framöver.

# Referenser


[1] Törn, Maria. Försvarsmaktens telenät- och markteleförband. *Försvarsmakten.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://www.forsvarsmakten.se/fmtm/.

[2] Projekt Prio. *Försvarsmakten.* [Online] Högkvarteret, Informationsstaben, den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://www.forsvarsmakten.se/sv/Om-Forsvarsmakten/Ekonomi-och-planering/Projekt-Prio/.

[3] Lipshitz, Raanan och Pras, Adi Adar. Not only for experts: Recognition-primed decisions in the laboratory. [red.] Henry Montgomery, Raanan Lipshitz och Berndt Brehmer. *How professionals make decisions.* New Jersey : Lawrence Erlbaum Associates, Inc, 2005. ss 91-93.

[4] Checkland, Peter and Holwell, Sue. *Information, systems and information systems - making sense of the field.* Chichester : John Wiley & Sons, Ltd, 1998. pp. 86-92. ISBN 0-471-95820-4.

[5] Eklund, Sven och Fernlund, Hans. *Programkonstruktion med kvalitet - projekthantering och ISO 9000.* Lund : Studentlitteratur, 1998. ss. 97-100. ISBN 91-44-00626-8.

[6] Enterprise Architecht. *Sparx systems.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://www.sparxsystems.com.au/.

[7] Functional requirement. *Wikipedia.* [Online] den 12 Maj 2011. [Citat: den 12 Maj 2011.] http://en.wikipedia.org/wiki/Functional_requirement.

[8] Non-functional requirement. *Wikipedia.* [Online] den 12 Maj 2011. [Citat: den 12 Maj 2011.] http://en.wikipedia.org/wiki/Non-Functional_Requirements.

[9] Active Directory. *Wikipedia.* [Online] den 13 Maj 2011. [Citat: den 13 Maj 2011.] http://sv.wikipedia.org/wiki/Active_Directory.

[10] Försvarsmakten. Direktiv för Försvarsmaktens informationsteknikverksamhet. 2004.

[11] Bengtsson, Johan och Hallberg, Jonas. *Test av relevans och validitet avseende sökerhetsvärdering - En studie av Försvarsmaktens metoder för säkerhetskravställning och sårbarhetsanalys.* Linköping: Totalförsvarets forskningsinstitut, 2008. Vetenskaplig rapport. ISSN 1650-1942.

[12] UML® Resource Page. *Unified Modeling Language.* [Online] den 13 Maj 2011. [Citat: den 13 Maj 2011.] http://www.uml.org/.

[13] Scott, Kendall. *The unified process explained.* Indianapolis : Pearson Education, Inc, 2002. ss. 19-36. ISBN 0-201-74204-7.

[14] Windows Forms. *Wikipedia.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://en.wikipedia.org/wiki/Windows_Forms.

[15] Web service. *Wikipedia.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://sv.wikipedia.org/wiki/Web_service.

[16] Globally unique identifier. *Wikipedia.* [Online] den 21 Maj 2011. [Citat: den 21 Maj 2011.] http://en.wikipedia.org/wiki/Globally_unique_identifier.

[17] SOAP. *Wikipedia.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://en.wikipedia.org/wiki/SOAP.

[18] ADO.NET Entity Framework. *MSDN.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://msdn.microsoft.com/en-us/library/bb399572(v=VS.90).aspx

[19] LINQ. *MSDN.* [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://msdn.microsoft.com/en-us/netframework/aa904594.

[20] Delegates (C# Programming Guide). *MSDN*. [Online] den 26 Maj 2011. [Citat: den 26 Maj 2011.] http://msdn.microsoft.com/en-us/library/ms173171(v=vs.80).aspx

[21] Implementing Singleton in C#. *MSDN*. [Online] den 22 Maj 2011. [Citat: den 22 Maj 2011.] http://msdn.microsoft.com/en-us/library/ff650316.aspx.

[22] Brehmer, Berndt. *Command and control as design*. Washington, DC. 2010. Proceedings of the 15th International Command ad control technology symposium.