Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik

Publiceringsdatum 2004-05-26

SMÅSKALIGA RFID-BASERADE INFORMATIONSSYSTEM

Utformning av ett generellt implementeringsverktyg

Abstrakt

RFID (Radio Frequency Identification) ger intressanta möjligheter att automatiskt, snabbt och på ett transparent sätt få in unika identiteter i ett informationssystem. Tillsammans med en databas ger RFID möjligheter att skapa ett informationssystem som i högre grad än med andra tekniker automatiserar datafångst och informationshantering. Att skapa ett sådant informationssystem kräver omfattande kunskaper. Höga kostnader för att realisera dessa kunskapskrävande tillämpningar innebär att merparten av befintliga system är storskaliga lösningar. Det här arbetet syftade till att ta fram en programvara som automatiserar delar av arbetet att göra småskaliga informationssystem grundade på RFID-teknik, och utnyttjar den funktionalitet som finns i vanliga databasprogram. För att reda ut hur en sådan programvara kunde utformas utvecklades prototyper som itererats och utvärderats av en grupp domänexperter. Resultatet av arbetet är en översiktlig beskrivning av hur en generell mellanliggande programvara som automatiserar sammankoppling och konfigurering av RFID-system och databas kan utformas. En slutsats av arbetet var att tillkomsten av småskaliga informationssystem med RFID-teknik underlättas genom den föreslagna mellanliggande programvaran.

Nyckelord: RFID, informationssystem, middleware, databas.

Författare: Jesper Hakeröd, Roland Thörner
Handledare: Faramarz Agahi
Magisteruppsats, 20 poäng

Innehållsförteckning

1. Introduktion

Ämnesområdet introduceras och vi argumenterar för att RFID är en intressant teknik när det gäller informationshantering. Vi ger också några exempel på informationssystem, både existerande och tänkta, som tar bruk av RFID-tekniken. Vidare introducerar vi en rad begrepp som återkommer i uppsatsen. Arbetets syfte liksom frågeformuleringen presenteras.

Tänk dig att du för första gången skall till att arrangera ett Vasalopp, året är 1995 och du har 12000 anmälda åkare. Ett arrangemang av det här slaget kräver en omfattande organisation, inte minst när det gäller resultathantering. Startfältet brakar iväg några tjuvstartsminuter före klockan 8 och drygt tre timmar senare kommer den förste åkaren till Mora. Därefter följer ett pärlband av åkare och för dem allesammans skall en sluttid samt ett 10-tal mellantider presenteras. Hur skall i storleksordningen 100.000 tider kunna avläsas och skrivas in i listor för alla åkare?

En möjlig lösning skulle kunna vara ett band med ett mycket litet elektronisk chip, som varje åkare sätter fast på smalbenet och som automatiskt klockar alla tider. Start-, mellan- och sluttid tas om hand av ett system som skriver in alla resultat direkt i anmälningslistan, utan fel. Alla tider görs omedelbart tillgängliga via Internet (Laadoe, 2003). Personal som skulle behövas för att klocka alla åkare, om det överhuvud taget varit möjligt, kan istället göra andra, viktigare saker. Blir du intresserad?

Den här uppsatsen handlar om informationssystem som använder sig av Radio Frequency Identification (RFID). RFID omfattar en rad delvis olika tekniker för trådlös kommunikation och identifiering av människor, varor, djur et cetera. Tekniken är mycket användbar i många sammanhang där det gäller att hålla reda på saker (Stanford, 2003). Det kan gälla idrottsutövare, odlad lax, kollektivtrafikresenärer, cementblandare, bildäck, bensinkunder eller något helt annat. Utgångspunkten är att det man vill hålla reda på har en unik identitet genom ett litet elektroniskt chip. Denna identitet kan läsas av automatiskt, trådlöst och med hög hastighet. Tekniken är inte heller beroende av ”fri sikt” utan kommunicerar genom många material (Finkenzeller, 1999).

Tekniken finns beskriven i litteratur och artiklar, bland böcker betraktas allmänt Finkenzellers (1999) verk ”RFID Handbook” som ett av de mest betydande. Mycket av det som skrivits om RFID är mycket tekniskt till sitt innehåll och det är slående hur lite som är skrivet kring nyttan, dvs. vilka värden den kan tillföra, mer än att man ofta fastslår att den har en mycket stor potential (Auto-ID Center, 2002). Finkenzeller (1999) beskriver ett antal befintliga användningsområden. En rad artiklar tar även upp möjligheter och tänkbara scenarios kring tekniken och dess framtida användningar, ofta ur ett relativt tekniskt perspektiv (Fildes, 2002). Ett annat område som diskuteras gäller eventuella hot mot den personliga integriteten (Borgström, 2004).

I dag används tekniken bland annat för märkning av djur: hundar, fiskar, nötkreatur, får m fl. (Tuttle, 1997). Tekniken används också industriellt, Assalub AB [1] har utvecklat ett smörjsystem som bygger på att man effektivt identifierar alla besökta smörjställen på exempelvis en pappersmaskin vid ett pappersbruk. Flera system för kollektivtrafik är också i bruk (Finkenzeller, 1999).

Flera av dem som reflekterar över utvecklingen av RFID i en nära framtid, (Ström, 2002; Raza et al, 1999) målar upp scenarier med storskaliga system, till exempel prismärkningssystem i varuhus. Auto-ID Center (2002) arbetar med systemet ”Savant” som närmast får liknas vid en global struktur för hantering av varor et cetera i ett sammanhang som kan omfatta många led, exempelvis tillverkare, distributör, handlare, kund med flera. Den här uppsatsen kommer att fokusera på småskalig användning av RFID. Med småskalig menas i sammanhanget exempelvis maskinuthyraren som vill skapa ett system för att identifiera alla sina uthyrningsobjekt eller sjukhuset som vill märka alla sina patientsängar för att hålla reda på all relevant data om dessa. Småskalig behöver dock inte innebära att det är få entiteter som skall hanteras.

Ett mer utvecklat exempel skulle kunna vara ett företag som hyr ut högtidskläder och märker alla plagg med en utifrån osynlig RFID-tagg (fr eng ”tag” = etikett, bihang, påhäng. Se avsnitt 1.1 begreppsdefinitioner). Det här innebär att hantering kring registrering av uthyrning och återlämning kan automatiseras. Kunden får snabbare service och uthyraren får bättre kontroll, bland annat så minskas risken att man tar fel plagg. Det är även lätt att kontrollera så att inget plagg bärs iväg utan att ha blivit registrerat. Samma tagg kan användas för sortering inför tvätt och hantera relevant data om plagget. Liksom att långa rader med plagg snabbt kan sökas av med hjälp av en handdator på efter jakt på rätt storlek och modell. I viss uthyrningsverksamhet kan tjänsten byggas ut med ett webbgränssnitt så att kunder hemifrån kan se vad som finns på lager och göra sina egna bokningar.

Många pekar på en stor expansion (Finkenzeller, 1999; Fildes, 2002) och ett möjligt genombrott för tekniken inom en snar framtid (Ryberg, 2004). Ström (2002) refererar till analysföretaget Frost & Sullivan som värderar världsmarknaden för RFID till 27 miljoner dollar år 2000 och förutspår att denna skall växa till det tiodubbla redan år 2004. En annan och tidigare prognos (Tuttle, 1997) förutspådde en marknad om 800 miljoner dollar år 2000 med en årlig tillväxt om 30-35 %.

Oavsett om den framtida tillväxten blir enligt de mest positiva prognoserna eller inte kan på goda grunder antas att det finns en potential för utveckling av både stora och småskaliga tillämpningar. Ett par betydande svårigheter i sammanhanget är bristen på programvara och att kostnaderna för tekniken ännu är relativt höga (Sarama, 2002; Takaragi et al, 2001; Howes et al, 1999).

Bristen på programvara kan antas innebära att oproportionerligt mycket tid får läggas på programutvecklingsarbete vilket spär på den redan höga kostnaden

för hårdvaran. Således finns en teknik med stor potential men där implementeringen är alltför komplicerad för att tillämpningar skall kunna realiseras av andra än ett ganska litet fåtal professionella utvecklare. Givetvis med höga kostnader som följd. Uppsatsen tar upp frågan om denna komplicerade hantering kan automatiseras och möjliggöra för fler att skapa den här typen av småskaliga tillämpningar.

1.1 Begreppsdefinitioner

Följande begrepp är centrala för det fortsatta läsandet av uppsatsen:

Tabell 1. Begreppsdefinitioner

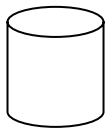
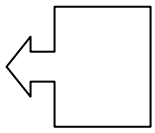
Användarvänlig	Används synonymt med ”usability” –begreppet. A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users (Bevan, 1999).
Applikation	”Ett program som dedicerar en dator till en särskild uppgift” (Beekman & Rathswohl, 2001, s. 22).
Auto-ID center	En industristödd organisation baserad vid Massachusetts Institute of Technology (MIT) i USA, University of Cambridge i UK samt vid University of Adelaide i Australien. Auto-ID Center utvecklar standarder för hård- och programvara kring RFID. En viktig funktion är att hantera de unika identiteter varje tag har (Auto-ID Center, 2002).
Databas	”…a computerized system whose overall purpose is to store information and to allow users to retrieve and update that information on demand” (Date, 2004, s. 6).
Databasprogram	Likställer vi med ett DBMS (Database Management System), ett program för att hantera data i en databas (Beekman & Rathswohl, 2001, s. 220).
Datafångst	Olika tekniker att läsa data som bärs av någon teknik, exempelvis streckkodsläsare eller RFID-mottagare, Mindscape [3].
Domänexpert	Person med djup kunskap inom det relevanta området (domänen).
Enkel databas	Ett enkelt databasprogram med vilket man relativt enkelt kan skapa mindre databaser och vilket många har tillgång till.
Entitet	”Ett objekt som skall representeras i en databas” (Date, 2004, s. 12).
Inbyggda system	Användande av elektronik i produkter för att göra dessa mer användarvänliga, mer funktionella, lönsamma et cetera, TeknIQ-projektet [4].
Informationssystem	"Ett IS är en samling komponenter, innehållande en struktur, som realiserar vissa funktioner, vilka förädlar materia eller information. Ett IS förändras över tiden och befinner sig i en viss miljö för att uppfylla vissa krav" (Apelkrans & Åbom, 2001, s. 20).
IT-artefakt	”Av människan skapad artefakt som har informationsteknik som bärande element i sin grundläggande struktur och funktionalitet” (Löwgren & Stolterman, 2000, s. 4).
Kvalitet	”alla sammantagna egenskaper hos ett objekt eller företeelse som ger dess förmåga att tillfredsställa uttalade eller underförstådda behov” (ISO 8402, 1994).
Läsare/antenn	(Kommer fortsättningsvis enbart att benämnas ”läsare”) Skickar ut en signal som uppfattas av taggen, vilken tar vara på en del av den utskickade energin, använder denna för sina egna interna operationer. För att därefter skicka sin identitet till läsaren. Läsare finns i en mängd olika utföranden för olika typer av ändamål och olika typer av taggar (Finkenzeller, 1999).

Makron	En möjlighet att automatisera en uppgift man utför ofta genom relativt enkel kodning.
Middleware/ Mellanliggande programvara	En sammanbindande programvara vars huvudsakliga syfte är att knyta samman två eller flera IT-artefakter och utnyttja deras sammanlagda funktionalitet.
ODBC	Open Database Connectivity (ODBC) är en standardisering av gränssnittet för applikationer mot databaser som pratar Structured Query Language (SQL).
RFID	Radio-Frecuency Identification. System för trådlös överföring av energi och data (Finkenzeller, 1999).
Skal	Ett program som utnyttjar en annan programvaras funktioner och döljer komplexiteten i den nödvändiga samverkan mellan mjukvarorna.
Skript	En delmängd av ett programspråk som kan användas för att skapa ny funktionalitet i exempelvis ett databasprogram.
Småskalig	Utvecklad med relativt knappa medel för användning i relativt begränsad omfattning .
Tag (tagg)/ transponder	(Kommer fortsättningsvis enbart att benämnas tagg) Taggen är den unika delen i systemet, taggen bär den alldeles särskilda identiteten som systemet identifierar. Taggar finns i en mängd utföranden och med olika karaktäristika (Finkenzeller, 1999).
Tillämpning	Praktisk användning av hård- och programvara i ett speciellt sammanhang.
Transparens	Underliggande system tar hand om detaljer i eller hela uppgifter för en användare så att han eller hon inte upplever procedurer utan enbart resultat.

Symboler

I uppsatsen förekommer figurer som visar uppbyggnad och/eller funktion av RFID-system:

Tabell 2. Symboler som används i uppsatsen

Databas	Läsare	Tagg
		

1.2 Syfte och problemformulering

RFID kan användas som en del i olika slag av informationssystem och det finns en stor användningspotential i tekniken (Tuttle, 1997; Raza et al, 1999; Sakamura, 2001; Sarma et al, 2001; Gyger & Desjeux, 2001; Ollivier, 1996).

Tre faktorer kan antas bromsa användandet. I första hand kostnader kring hårdvara men också delvis brist på standarder och generell programvara. Detta styrks av såväl våra intervjuer samt artiklar (Tuttle, 1997; Raza et al, 1999). Steget mellan att se en taggs identitet i exempelvis en applikation som Hyper Terminal till att plocka in samma identitet i en databas, eller att få till stånd en tillämpning är långt, både tids- och kompetensmässigt.

Argumentation kan föras för att RFID-tillämpningar inte skiljer sig från andra där data eller identiteter registreras, exempelvis ett streckkodssystem. RFID-tillämpningar skiljer sig från andra system för datafångst på några punkter, bland andra: (Ström, 2002, s. 176):

- Mängden entiteter, ett RFID system kan innehålla miljontals entiteter.
- Mängden entiteter per tidsrymd som skall hanteras kan uppgå till många i sekunden, eller tiotusentals per timma.
- Transparensen för användaren.

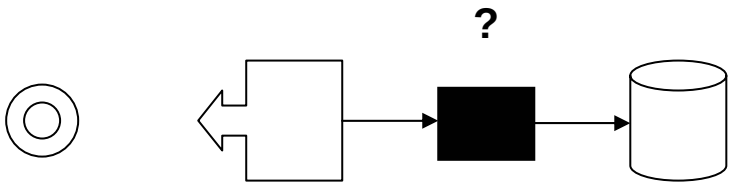
För att belysa omfattningen av arbetet att skapa ett sammanhängande system mellan RFID och en databas hänvisas till bilaga 6 där två enkla referenstillämpningar presenteras.

En bärande tanke i uppsatsen är att systemet skall vara enkelt att använda. Den som har grundläggande kunskaper om RFID och dessutom kan skapa en enkel databas borde också kunna skapa ett sammanhängande system av detta.

Syftet med uppsatsen är att undersöka hur en generell mellanliggande programvara kan utformas för att underlätta tillkomsten av småskaliga RFID-tillämpningar. Resultatet av arbetet blir ett förslag som övergripande beskriver en sådan mellanliggande programvara.

Problemformuleringen lyder:

Hur kan en generell, användarvänlig programvara utformas för att koppla samman RFID-identiteter med en databas, och därigenom underlätta skapandet av småskaliga RFID-baserade informationssystem?



Figur 1: Grafisk översikt av problemformulering.

Resultatet av arbetet kommer att bli ett systemförslag på en sådan mellanliggande programvara eller om man hellre vill använda uttrycket ”brygga mellan RFID och databas”. Fortsättningsvis kommer den att benämnas mellanliggande programvara.

1.3 Avgränsningar

Uppsatsens fokus ligger på småskaliga tillämpningar, begreppet småskalig skall förstås som en tillämpning utvecklad med relativt sett begränsade resurser. Småskalighet definieras i avsnitt 5.4.

Det finns en rad olika typer av taggar med varierande grad av funktionalitet, aktiva taggar, taggar som krypterar trafik, skrivbara taggar, taggar som kan hantera kollisioner et cetera. Uppsatsen avgränsas till att beröra enbart de taggar som minst klarar av att uppge en unik identitet, då dessa kan identifieras unikt i ett databassystem och således kan representera en entitet (Date, 2004; Ullman & Widom, 2002). Vårt systemförslag kommer inte att stödja inmatning till read-and-write taggar.

I arbetet har två referenstillämpningar skapats. Dessa tjänar flera syften och de viktigaste är:

1. De ger en bild av arbetsinsatsen att skapa en mycket begränsad tillämpning.
2. De beskriver delar av den mellanliggande programvarans funktionalitet.

Referenstillämpningarna gör inte anspråk på mer än att hantera kommunikation mellan RFID och databas, och visa mycket enkel funktionalitet i en tillämpning. Detta innebär bland annat att säkerhetsmässiga och andra överväganden inte gjorts.

Som teoretisk bas i intervjuer med domänexperter används begreppet Quality in Use (Bevan, 1999). Bevans modell omfattar flera nivåer varav de högsta är user och work environment. Eftersom inte systemet realiseras och implementeras kan inte dessa nivåer behandlas fullt ut. Se vidare avsnitt 2.3.

I uppsatsarbetet har prototyping använts som systemutvecklingsmetod. Prototypen kommer att ha viss funktionalitet. Syftet är att övergripande beskriva ett förslag på en generell mellanliggande programvara, inte att realisera det.

2. Teori

Kapitlet tar upp teoriområden kring inbyggda system, möjliggörande tekniker för inbyggda system som är relevanta för det här arbetet, nämligen databaser, ODBC och RFID-tekniken. Vi går därefter vidare och redogör för de teoretiska begrepp som ligger till grund för den delen av vårt arbete som är mer systemutvecklingsinriktat, Quality in Use, prototyping, kravprocessen och en lite djupare definition av begreppet småskalighet.

2.1 Inbyggda system

Vid Sektionen för Informationsvetenskap, data- och elektroteknik vid Högskolan i Halmstad finns CERES – Centre for Research on Embedded Systems. CERES är ett ledande forskningscenter inom området inbyggda system med speciell inriktning mot ”Cooperating embedded systems”. Sektionen är vår arbetsplats och detta stimulerar oss i såväl val av undersökningsområde som sätt att se på undersökningsområdet.

Med inbyggda system menar vi vardagligt olika slag av elektronik som vi sätter in i produkter för att ge dessa nya och förbättrade egenskaper. Wolf (2002, s. 136) definierar begreppet som ”en dator som är en komponent i ett större system och som är beroende av sin egen mikroprocessor”. En annan gängse definition på inbyggda system är den som förs fram inom KK-stiftelsens expertkompetensprogram, TeknIQ-projektet [4], där man avser:

> *”Användande av elektronik i produkter för att göra dessa mer användarvänliga, mer funktionella, lönsamma et cetera”.*
>
> [4]

Pär Ström (2002) för fram begreppet M2M, som vanligen står för ”maskin till maskin” men som han definierar vidare än så. Han menar att M2M kan stå för olika kombinationer av orden produkt och människa, exempelvis Produkt till Människa och där minst en produkt som inte är en dator måste vara inblandad.

CERES (Svensson et al, 2004) diskuterar begreppet som

> *”...embedded electronics for computing and communicating...”.*
>
> (s.5)

Av detta följer att oberoende om vi talar om embedded computing, inbyggda system, embedded systems eller M2M så faller de RFID-system den här uppsatsen handlar om under begreppen, även om det tillhör en mera enkel form (Svensson, 2004). Skälen till att använda inbyggda system är många. Ström (2002) för fram i huvudsak ekonomiska argument. TeknIQ-projektet [4] hävdar också ekonomiska aspekter men trycker också på att produkter kan tillföras ny och bättre funktionalitet, bättre användarnytta, bättre gränssnitt samt att digitala produkter lätt kan kommunicera med andra digitala produkter. Det här är

även CERES huvudspår, inbyggda system står inför ett paradigmskifte. Från att ha varit ganska isolerade system ser man en utveckling mot system som kommunicerar och samverkar (Svensson et al, 2004). Per definition är ett inbyggt system alltid en del av något annat, och det tillför eller förbättrar något som detta andra inte har själv.

Så här långt har vi rört oss på en ganska konceptuell nivå i vår definition. För att närma oss uppsatsens fokus behöver vi också se vad ett inbyggt system är på en mer konkret nivå. För ändamålet är Wolfs (2002) ovan nämnda definition en bra utgångspunkt, när han talar om en komponent i ett större system som har sin egen intelligens.

Det kan exempelvis vara en Internetansluten fräsmaskin i industrimiljö, där det inbyggda systemet är en webbserver med anslutning till svarvens egna kontrollsystem, exempelvis en PLC (Programmable Logical Controller) och där tekniken möjliggör att man kan ”surfa in i svarven”, dvs man kan avläsa status, uppdatera programvara, övervaka, eller om säkerheten medger, köra svarven från vilken Internetansluten dator som helst. I ett sådant system kan alla frässtål vara RFID-märkta för att förhindra att man kör maskinen med fel varvtal eller belastning. Maskinen kan själv korrigera sitt beteende beroende på informationen som finns lagrat om varje unikt stål (Finkenzeller, 1999).

CERES fokus på den här typen av samverkande inbyggda system är särskilt intressant relativt den utveckling många förutspår när det gäller användandet och omfattning i sättet att bruka RFID. Sakamura (2001) skriver att Internet de senaste 10 åren blivit en viktig infrastruktur i världen och förutspår nu att

> *“ubiquitous och pervasive computing is likely to have an impact of a similar magnitude as that of the Internet. Computers embedded in everyday objects will communicate and cooperate with each other to enhance the services of formerly standalone single devices”.*
>
> (s. 4)

Ett genomslag som Sakamura förutspår kan skapa mycket stora behov av småskaliga tillämpningar inom RFID-området då det på mängder av ställen, exempelvis en mekanisk verkstad, kan finnas taggar och möjlighet att skapa tjänster kring dessa. Tekniken kan med andra ord bana vägen för en tjänst av det slag som Dahlbom beskriver:

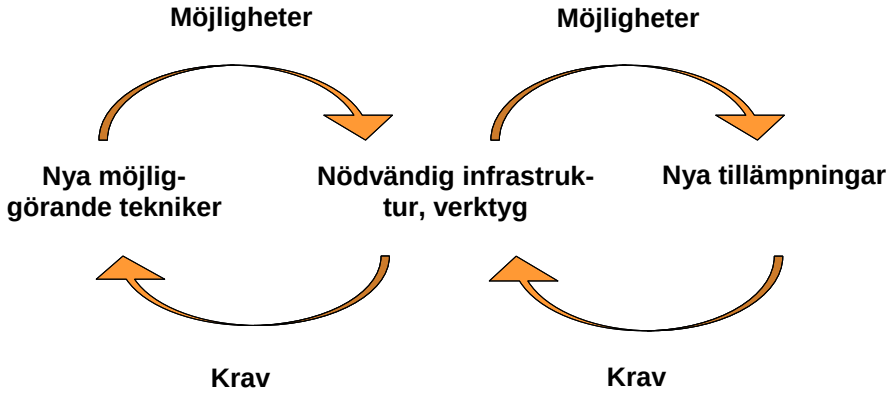
> *“ Were we choose a concept today, in the early 2000s, to characterize the use of information technology in organizations and everyday life, systems is not the one we first would come up with. Systems make way for services, and we speak of information services, networked based ser vices…”.*
>
> (Dahlbom, 2001, s. 1)

En viktig distinktion är att RFID systemet är ett subsystem av något annat. RFID kan mot den här bakgrunden inte ses som enbart en elektronisk etikett, utan kan bli en del av ett väsentligt mycket större system, där den utför sin uppgift. I sin bäst fungerande form är dessa subsystem helt transparenta för användaren, dvs när man surfar in i fräsmaskinen ser man inga taggar utan frässtål med olika egenskaper. Ett exempel på ett mer renodlat transparent RFID-system är EasyRide (Gyger & Desjeux, 2001), ett betalsystem för kollektivtrafik. Passageraren går bara ombord på bussen eller tåget och systemet debiterar resenären för den sträcka som åks, passageraren upplever en tjänst.

För vårt arbetes del får transparensen konsekvenser i utformningen av den mellanliggande programvaran. I ett automatiskt och transparent system måste alla komponenter utformas för att stötta denna automatik och transparens.

2.2 Möjliggörande tekniker

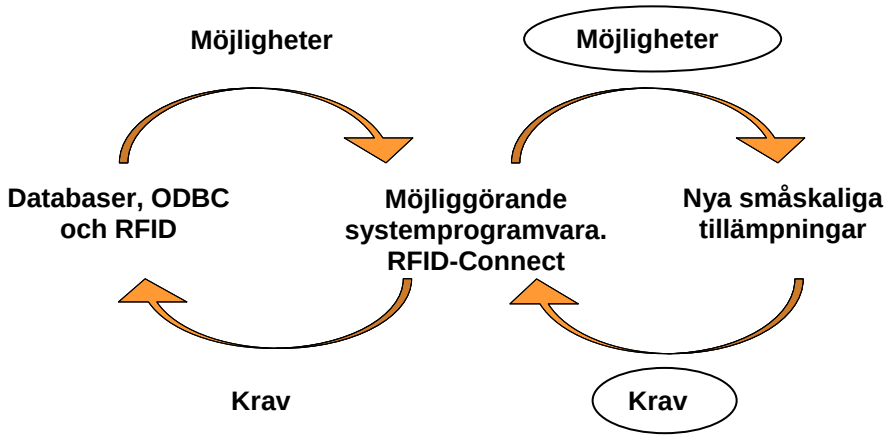
CERES inriktning mot samverkande inbyggda system innebär enligt Svensson (2004) att man riktar sitt intresse mot en nödvändig infrastruktur mellan nya möjliggörande tekniker (enabling technologies) och nya tillämpningar (new applications). För att kunna utveckla denna infrastruktur måste man ha kunskap om de möjliggörande teknikerna, hur önskemålen om nya tillämpningar kan se ut och utifrån detta skapa den nödvändiga infrastrukturen (co-operation solutions) vilken kan vara ett verktyg, ett protokoll, en hårdvarulösning, en möjliggörande systemprogramvara et cetera. Den nödvändiga infrastrukturen möter och skapar krav, liksom tar vara på och skapar möjligheter från nya möjliggörande tekniker och nya tillämpningar.



Figur 2. CERES-glasögonen. En brygga mellan teknik och tillämpning. Efter Svensson et al (2004, s. 7) som kompletterats med möjligheter och krav utifrån samtal med Bertil Svensson (2004).

Nya möjliggörande tekniker kan vara både av slaget nya tekniker som exempelvis RFID, och kompletterande tekniker som sedan länge är etablerade, som exempelvis databaser. Nya tillämpningar kan vara ett informationssystem som drar nytta av RFID-teknikens fördelar. Den nödvändiga infrastrukturen kan vara ett verktyg som underlättar tillkomsten av nya tillämpningar. Kraven från den nödvändiga infrastrukturen kan vara förbättrad funktionalitet i ett data-

basprogram. Möjligheterna som möjliggörande tekniker öppnar kan vara snabb och säker identifiering via RFID (Svensson, 2004).
Vi har gett den nödvändiga infrastrukturen, vårt planerade systemförslag, arbetsnamnet RFID-Connect. Gällande nya tillämpningar så hänvisar vi till de exempel som ges på olika ställen i uppsatsen. De möjliggörande teknologierna presenteras senare mera ingående under 2.2.1 – 2.2.3.



Figur 3. CERES-glasögonen (Svensson et al, 2004, s. 7; Svensson, 2004) modifierad med för uppsatsen aktuella begrepp.

Av modellen framgår att den mellanliggande programvaran ger vissa möjligheter, vilka kan sägas representera en del av syftet med uppsatsen, att skapa förutsättningar för nya småskaliga tillämpningar.

Den andra intressanta iakttagelsen är att den mellanliggande programvaran kommer att underställas krav. Vilka är dessa? En del av den frågan representeras av problemformuleringen. För att förstå vilka kraven är kommer vi att utgå från Bevan (1999) och den teori han för fram genom Quality in use.

För den här uppsatsens vidkommande är den intressanta iakttagelsen att arbetet inte sker i något vakuum utan i ett samspel eller spänningsfält mellan nya tillämpningar och möjliggörande teknik. Notera att modellen är relativ, begreppen är med andra ord alltid relativt de andra delarna i modellen. Ett och samma begrepp kan med andra ord beroende på sammanhang, sorteras in på olika ställen. Den intressanta iakttagelsen är dock inte denna rörlighet bland olika objekt i den teoretiska modellen utan att det krävs vissa förutsättningar för utveckling och att det finns en inneboende dynamik kring en programvara som den som beskrivs.

Följande avsnitt kommer att introducera de möjliggörande tekniker eller fundament som den mellanliggande programvaran bygger på. De begrepp som är aktuella är RFID, ODBC och Databas.

2.2.1 Databas

> *"…a database system is basically a computerized recordkeeping system ;in other words, it is a computerized system whose overall purpose is to store information and to allow users to retrieve and update that information on demand".*
>
> (Date, 2004, s. 6)

Date (2004) beskriver en databas som ett datoriserat system för att lagra information och låta användarna hämta och uppdatera data. För att hantera denna information använder vi ett enligt Beekman & Rathswohl (2001) ett databasprogram. Det vi bland annat uppnår med en databas, utifrån Beekman & Rathswohl (2001), är möjlighet att:

- Spara stora mängder information.
- Söka bland denna information snabbt och korrekt.
- Sortera och organisera information på olika sätt.
- Skriva ut och distribuera information på olika sätt.

Informationen relateras till en primärnyckel (Date, 2004) och denna eller något annat fält i varje post kan unikt knytas till en RFID (Auto-ID Center, 2002) (se bilaga 7). På detta sätt kan reella objekt (människor, videofilmer, hundar, frässtål et cetera) relateras till en unik post i en databas med hjälp av en RFID-tagg. Genom att använda ett relativt enkelt databasprogram öppnas då stora möjligheter att använda detta för olika tjänster.

Databasen är en möjliggörande teknik i sammanhanget och så långt som möjligt kommer dess funktionalitet tillsammans med makron och skript att användas för funktionaliteten i hela systemet. Detta ger exempelvis möjligheter att (egen slutsatser utifrån Conolly & Begg, 2002) :

- Tidsstämpla data vilket innebär att man kan använda systemet för timdebitering, godkänd in- och utpassering med behörighetskontroll.
- Producera olika formulär som möjliggör automatisk utskrift av exempelvis en faktura grundad på vilka taggar som lästs av, eller en bekräftelse på att något är inlämnat för reparation.
- Kontrollera mot saldon, vilket innebär att systemet larmar om antalet komponenter av ett visst slag underskrider ett minimivärde efter det att en entitet plockats ur systemet.
- Göra data tillgänglig på Internet vilket innebär att data kan presenteras via hemsida, det kan exempelvis gälla lediga platser på en campingplats eller liknande. Eller göra databasen sökbar för behörig användare där man på distans kan kolla sådant som lagersaldon och status.
- Exportera data till andra applikationer, det kan vara ett styrsystem, en ordbehandlare, en annan databas, ett ekonomisystem, en elektronisk skylt och så vidare.

- Exportera data till andra plattformar, trådlöst till en handdator, en mobiltelefon, till en dator med annat operativsystem et cetera.
- Naturligtvis rutinåtgärder som att bokföra att någon har lånat, hyrt eller köpt något. Vid lån kan återlämningskontroll enkelt automatiseras. Vid hyra kan ett avtal skrivas ut och vid köp ett kvitto.

2.2.2 ODBC (Open database connectivity)

När en programvara behöver kommunicera med ett databashanteringssystem behöver först en anslutning upprättas, därefter kan kommunikationen ske. Open database connectivity (ODBC) är en standard som tillhandahåller ett sk application programming interface (API). Ett sådant API tillåter program att anropa databashanteringssystemet (DBMS) förutsatt att programvaran som behövs för aktuellt DBMS är installerad. Det finns drivrutiner för de flesta DBMS på marknaden vilket gör det enkelt att välja det DBMS som önskas (Elmasri & Navethe, 2004).

Elmasri & Navethe (2004) presenterar tre olika sätt att ifrån en applikation kommunicera med en databas; genom (1) inbäddad structured query language (SQL), (2) genom att använda funktionsbibliotek eller genom att (3) skapa ett helt nytt språk:

1. Inbäddad SQL innebär att t ex SQL- uttryck bäddas in i koden med ett särskilt prefix vilket i sin tur innebär att den kod som är databasspecifik kan lyftas ut vid kompilering av applikationen . Den databasspecifika koden kan kompileras separat för att därefter anropas via funktioner ifrån applikationen och om dessa frågor behöver ändras krävs dock att applikationen kompileras om (Elmasri & Navethe, 2004).

Date (2004) påtalar behovet av kunna skapa SQL-frågor under tiden en applikation exekveras, genom sk dynamisk SQL, vilket även styrks av Elmasri & Navethe (2004). Enligt Date (2004) kan om en applikation bara behöver ett fåtal SQL-frågor, dessa hårdkodas i koden. Ofta kan SQL-frågorna vara många och av väldigt varierande karaktär vilket Date (2004) menar skapar ett behov av att kunna konstruera SQL-frågor under tiden applikationer exekveras. Detta beskrivs av Elmasri & Nevethe (2004) genom ett exempel där en användare dynamiskt kan konstruera ett SQL-uttryck genom att i peka och klicka i gränssnittet för att t ex lista data ifrån en databas.

2. Funktionsbibliotek, application programming interface (API), är enligt Elmasri & Navethe (2004) är ett annat sätt att kommunicera med databaser. Date (2004) och Elmasri & Navethe (2004) lyfter fram tekniken SQL Call-Level Interfaces (CLI) och Open Database Connectivity interface (ODBC).

Enligt Date (2004) finns två skäl som talar för SQL CLI och ODBC gentemot dynamisk SQL. För det första att dynamisk SQL bäddas in i källkoden och kräver att den kompileras under körning innan de kan användas. SQL CLI och ODBC

använder sig av särskilda funktionsanrop anpassade för den databas som används vilket gör att ingen särskild databasspecifik kompilering behövs utöver standardspråkskompileringen. För det andra är CLI och ODBC oberoende av vilket databashanteringssystem som används, med andra ord går det i princip bra att använda vilket databashanteringssystem som helst till en applikation till skillnad från att skriva SQL-frågor för ett specifik databashanteringssystem (ibid). Elmasri & Navethe (2004) påpekar att den kontroll som en förkompilering av SQL-uttryck utgör måste göras med funktionsbibliotek under exekvering av applikationen. Hur som helst är användningen av funktionsbibliotek det mest lämpliga sättet att kommunicera med databaser i en mellanliggande programvara eftersom vi inte behöver skräddarsy SQL-uttryck för ett specifikt databashanteringssystem.

3. Att skapa ett helt nytt språk är också ett alternativ, men det är knappast något som är intressant i samband med utvecklingen av en mellanliggande programvara för en småskalig användare.

Vidare beskriver Conolly & Begg (2002) ODBC som en teknik som ger en universal dataåtkomst i SQL databaser och fördelen med en generellt skriven applikation som har möjlighet att komma åt data i olika databashanteringssystem (DBMS), t ex MySQL, Access, Oracle, etc. Detta ger möjlighet för utvecklare att bygga och distribuera applikationer (t ex i client-servermiljö) utan att ta hänsyn till vilket specifikt databashanteringssystem som används, vilket som vi antytt tidigare lämpar sig väl för en mellanliggande programvara som RFID-Connect.

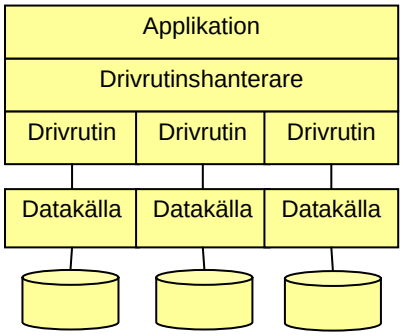
Det finns många alternativa tekniker för att kommunicera med SQL databaser som växt fram och anpassats för bl a internet. Conolly & Begg (2004) beskriver JDBC, OLE DB, DAO, COM+, ADO och .NET för att nämna några av dessa tekniker. Enligt Conolly & Begg (2002) har de flesta av dessa tekniker möjlighet att nå en ODBC stödjande databas. Det finns alltså många olika tekniker tillgängliga för att kommunicera med databaser och många av dem använder sig av ODBC. Detta förstärker vårt intryck av att ODBC är en beprövad och mycket lämplig teknik att använda sig av vid i samband med denna typ av informationssystem som inkluderar RFID-tekniken.

Conolly & Begg (2002) definierar ODBC arkitekturen som:

- Ett bibliotek av funktioner som tillåter applikationer att koppla upp sig mot DBMS, exekvera SQL kommandon och erhålla resultat.
- Ett standardiserat sätt att koppla upp sig mot ett DBMS.
- En standardrepresentation av olika datatyper.
- En standarduppsättning av felmeddelanden (koder).
- SQL syntax baserad på X/Open och ISO Call CLI specifikationer.

Vidare består en ODBC-arkitektur av fyra huvudsakliga komponenter enligt Conolly & Begg (2002), nämligen applikation, drivrutinshanterare, drivrutin

och datakälla, se figur 4. Applikationen processar kod och gör anrop genom att anropa ODBC-funktioner för att kunna köra SQL uttryck mot en DBMS och därigenom erhålla något resultat. Drivrutinshanteraren kan processa anropen av ODBC-funktionerna eller skicka dem vidare till rätt drivrutin. Drivrutinen processar ODBC-funktionsanropen och skickar iväg SQL frågor till rätt datakälla för att erhålla resultatet åt applikationen. Datakällan innehåller den data som applikationen vill ha åtkomst till och den associerade databasen (ibid). Detta är vad installationsdelen i den mellanliggande programvaran, RFID-Connect, lägger grunden för. Med andra ord ser RFID-Connect till att den har rätt drivrutin, drivrutinshanterare och datakälla för att kommunikationen mot databasen ska fungera.



Figur 4: ODBC-arkitektur för flera drivrutiner enligt Connolly & Begg (2002, s 678).

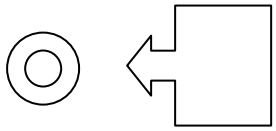
Elmasri & Navethe (2004) beskriver inte arkitekturen som Date gör utan snarare processen som behövs när en programmerare behöver åtkomst till en databas. Den beskrivs som en typisk sekvens bestående av tre steg. (1) Först måste applikationen etablera en koppling till databasen, t ex genom en Internetadress samt användarnamn och lösenord för åtkomst till databasen. (2) När en koppling är upprättad kan därefter applikationen kommunicera med databasen via exempelvis SQL-uttryck. (3) Till sist stängs kopplingen när kommunikationen med databasen inte längre behövs. Detta förfarande speglas tydligt i samband med våra prototyper.

Utifrån det resonemang som förts ovan kan det konstateras att ODBC är en teknik som lämpar sig väl för applikationer i den småskaliga verksamheten, vilken ger användaren möjlighet att använda nästan vilket databashanteringssystem som helst, inte bara för vanliga system utan även när det gäller system som görs tillgängliga via Internet.

2.2.3 RFID (Radio Frequency Identification)

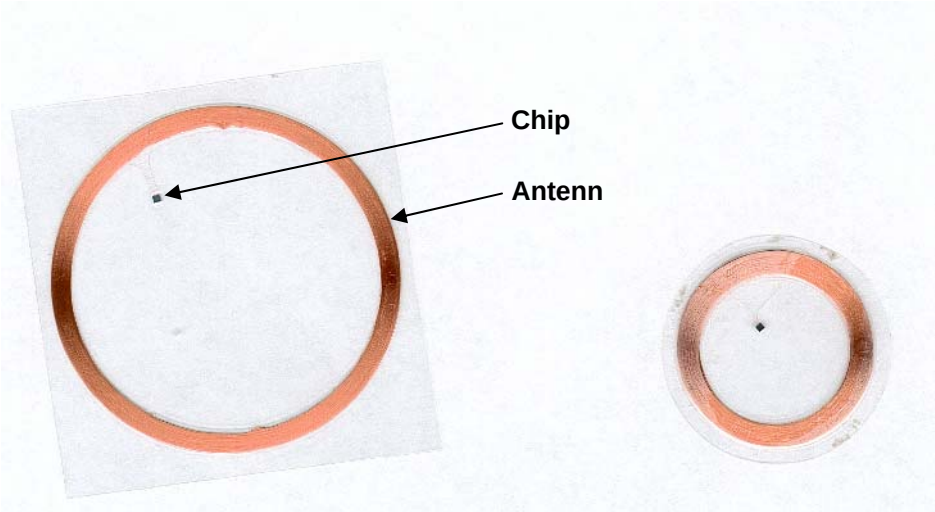
Ett RFID system består enligt Finkenzeller (1999) alltid av två komponenter:

- Taggen, som finns på objektet som skall identifieras.
- Läsaren, som läser data från taggen.



Figur 5. Schematisk bild av RFID system utifrån Finkenzellers (1999) definition.

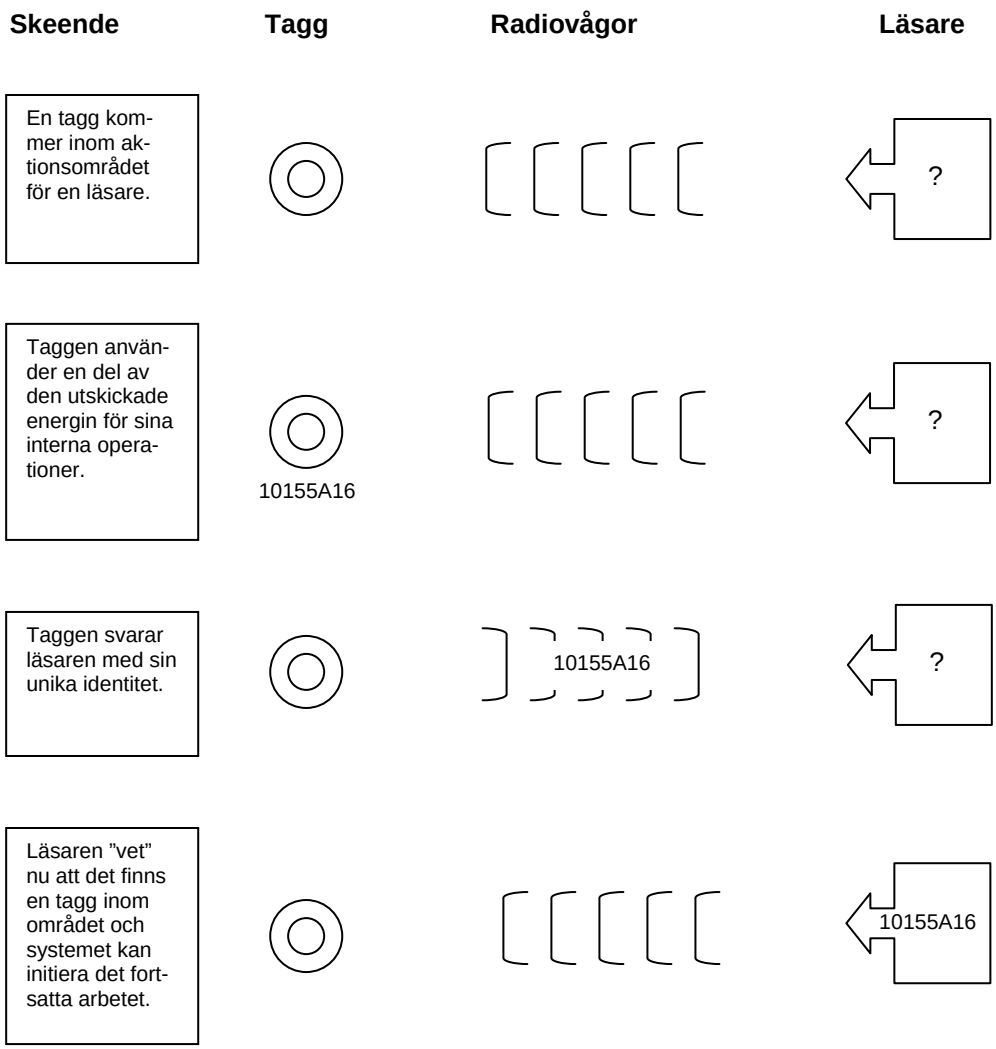
Taggen består av en liten (ner till 0,4 x 0,4 mm) elektronisk krets (Finkenzeller, 1999; Ström, 2002; Takaragi et al, 2001). Till denna krets är kopplat en antenn som utgör en del av taggen. Taggen är helt passiv och har ingen egen strömförsörjning. (Jfr dock ”aktiva taggar”). Beroende på grad av teknisk komplexitet kan taggen utföra olika saker som att kryptera data, hantera kollisioner, et cetera.



Figur 6. Exempel på taggar. Okapslat utförande. Naturlig storlek.

Läsaren är den andra av Finkenzellers två definierade delar. Från denna skickas radiovågor ut. När en tagg kommer inom läsarens aktionsområde tar taggen till vara på en del av den utskickade energin, använder denna för sina interna operationer, och svarar med just sin unika identitet, vilken då görs tillgänglig för systemet (Finkenzeller, 1999; Auto-ID Center, 2003).

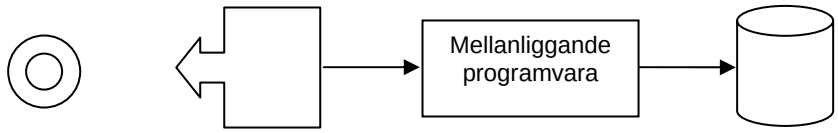
Figur 7. Exempel på läsare (stationärt utförande ca 80x120 mm).



Figur 8. Starkt förenklad översikt av RFID-system i verksamhet.

Med en applikation som exempelvis Hyper Terminal, är det mycket enkelt att läsa en taggs identitet. För att RFID-tekniken ska kunna användas till något

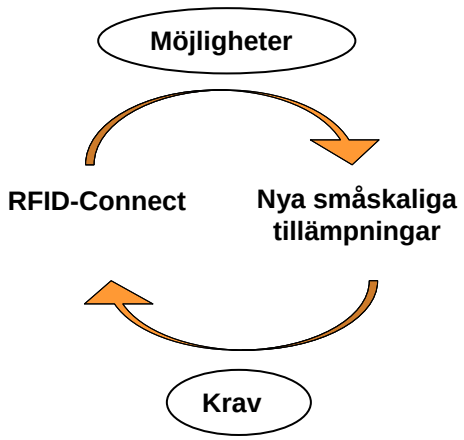
meningsfullt behövs även någon information om objektet som taggen följer, det kan gälla saker som namn, storlek, ägare, tvättanvisningar, serviceintervall mm. Ett RFID system behöver därför bestå av ytterligare en del, en databas i vilken informationen lagras enligt Ahlkvist Scarfeld (2001). RFID-identiteten som kommer in i datorn från RFID-läsaren via någon av datorns portar måste göras tillgänglig för databasen. Detta sker genom någon form av programvara. Vi definierar således ett RFID-baserat informationssystem som ett system av RFID-taggar och –läsare, mellanliggande programvara och databas.



Figur 9. Schematisk översikt av RFID-baserat informationssystem.

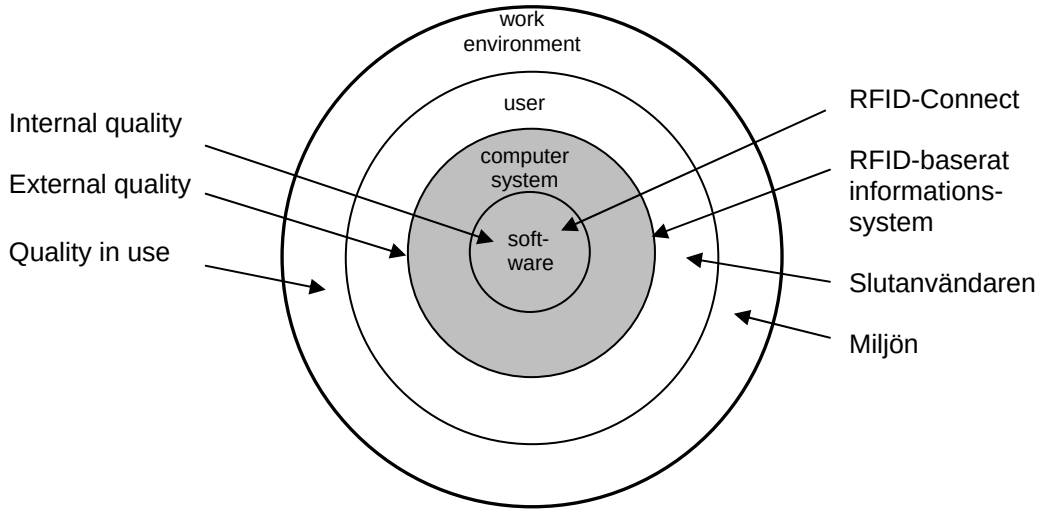
2.3 Quality In Use

I kapitel 2.2 konstaterades att en mellanliggande programvara kommer att underställas krav från de nya tillämpningarna och deras användare. Kraven omfattar alla de krav som normalt ställs på mjukvaror som användarvänlighet, funktionalitet et cetera, plus tillkommande krav på transparens. Målet med uppsatsen är att beskriva ett verktyg, en mellanliggande programvara som knyter samman tre IT-artefakter, RFID, ODBC och databaser och skapar underlag för enklare tillkomst av RFID-baserade informationssystem. För ändamålet kommer en prototyp att skapas vilken presenteras för en grupp domänexperter. Prototypen underkastas under itereringen löpande utvärdering. För denna utvärdering med domänexperterna utgår vi från Bevan (1999) och begreppet Quality in Use.



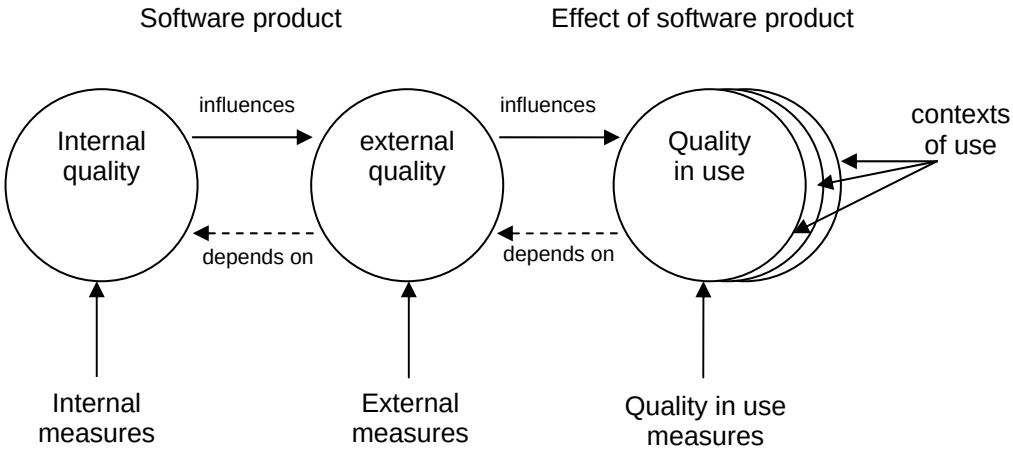
Figur 10. Iteration av prototyp där användarkraven utvärderas efter Quality in Use. Figuren är delvis återgiven och modifierad utifrån CERES-glasögonen (Svensson et al, 2004, s. 7; Svensson, 2004).

”Quality in use” (QIU) (Bevan, 1999) mäts efter i vilken utsträckning programvaran möter användarens behov i miljön där den används. Enligt Bevan finns det tre nivåer i det något bredare quality-begreppet: internal quality, external quality och QIU. Förenklat kan sägas att internal quality representeras av en applikations mest innersta väsen, exempelvis koden den är skriven i. External quality säger något om hur applikationen beter sig när den körs och när det gäller QIU ser vi också på hur applikationen möter de behov den är satt att lösa i en specifik situation. QIU-begreppet är enligt Bevan (1999) användarens syn på kvalitet när hon utför representativa uppgifter i en realistisk arbetsituation.



Figur 11. Relation mellan internal quality, external quality och quality in use. Efter Bevan (1999, s. 90). Uppsatsens begrepp harkompletterats och placerats in till höger i figuren.

Begreppet Quality in Use omfattar som vi ser i figur 11 även användaren och miljön (user respektive work environment). Eftersom den mellanliggande programvaran inte kommer att realiseras eller implementeras berörs inte de områdena i uppsatsen. Ett viktigt påpekande är dock att internal och external quality påverkar möjligheterna att uppnå Quality in Use, se figur 12.



Figur 12. Approaches to software product quality efter Bevan (1999, s. 90).

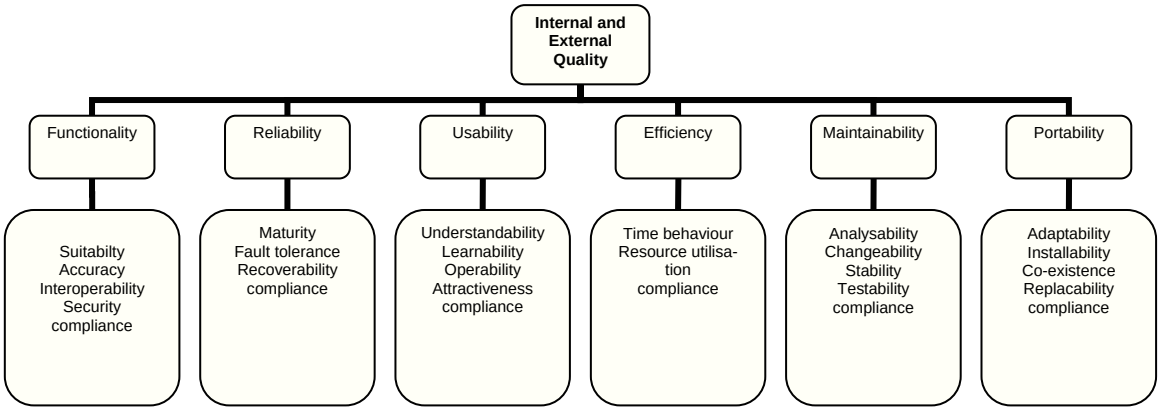
För att skapa så goda förutsättningar som möjligt för den mellanliggande programvaran att uppfylla de krav tänkta användare har, kommer ett antal domänexperter att intervjuas kring de begrepp som ställs upp i QIU. Quality in Use utgör således den teoretiska ramen för prototyp och systemutvärderingen.

Gällande external quality, enligt Bevan (1999), ser man till hur programvaran genom sin internal quality samspelar med systemet i övrigt, dock utan att ta med faktorer från omgivningen. Inom det här området finns de mer generella frågorna kring den mellanliggande programvaran. ISO/IEC 9126-1 (1998) definierar en rad begrepp kring programvara och ger oss en ram för att mäta och utvärdera external quality:

Tabell 3. Definitioner enligt ISO/IEC 9126-1.

Functionality	The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
Reliability	The capability of the software to maintain its level of performance when used under specified conditions.
Usability	The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
Efficiency	The capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.
Maintainability	The capability of the software to be modified. Modification may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specification.
Portability	The capability of software to be transferred from one environment to another.

Ovanstående huvudegenskaper bryts enligt ISO/IEC 9126-1 (1998) ner i underegenskaper som visas i figur 13.



Figur 13. Intern och extern kvalitet med sex huvudgrupper och 27 underegenskaper enligt ISO/IEC 9126-1 efter Håkansson (2000, s. 18).

Hittills har mest de två innersta cirklarna i figur 11 beaktats, de två återstående är ”user” och ”work environment” och är de två resterande cirklar som måste tas hänsyn till för att vi skall kunna prata om QIU.

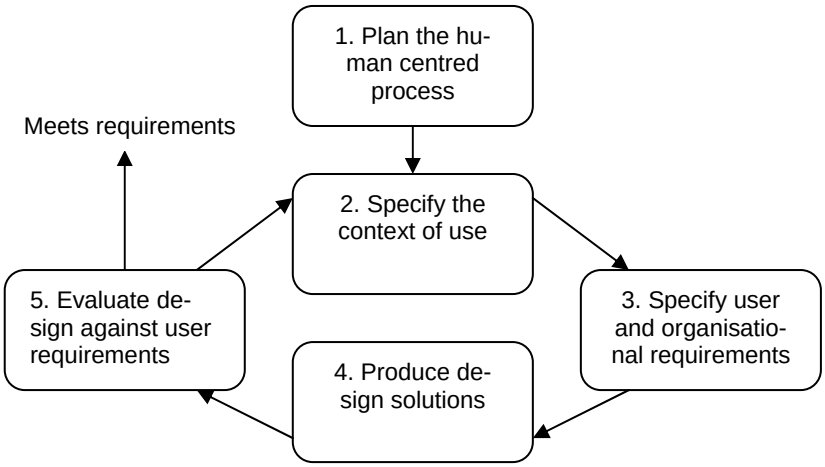
> *"Quality in use is the user's view of the quality of a system containing software, and is measured in terms of the result of using a software, rather than properties of the software itself. Quality in use is the combined effect of the software quality characteristics for the user".*
>
> (Bevan, 1999, s. 91)

Det finns enligt Bevan (1999) flera skäl till att designa för QIU:

- Increased efficiency.
- Improved productivity.
- Reduced errors.
- Reduced training.
- Improved acceptance.

Bevan reflekterar över vilka hinder som kan finnas för att inte nå de uppsatta målen kring design av programvara. För att överbrygga hindren krävs enligt Bevan en användarcentrerad systemutveckling ("user-centred approach to design").

Flera EU-finansierade projekt har utvecklat modeller för användarcentrerad systemutveckling som erbjuder lösningar på kulturella, tekniska och strategiska hinder. Bland projekten nämner Bevan (1999, s.93) MUSiC (1994), MAPI (1997), INUSE (1998) och RESPECT (1998). Stegen i utveckling som Bevan (1999) förespråkar är:



Figur 14. User centred design activities (Bevan, 1999, s. 94).

I Bevans modell syns en klassisk iteration, dvs designen drivs runt med gradvis förbättring till dess vi har skapat något som möter de krav vi formulerat. Bevan säger inte mycket mer om hur man utvecklar ett system som möter dessa krav. Däremot konstaterar han att INUSE och RESPECT projekten har satt upp egna nätverk av "Usability Support Centres" som erbjuder hjälp till organisationer

som saknar egen kompetens inom området. INUSE [2] erbjuder exempelvis ”Handbook of User-Centred Design” .

2.4 Prototyping

Enligt Bevan (1999) ger användarcentrerad systemutveckling bästa förutsättning att utveckla system som uppfyller kraven för QIU. Ett vanligt sätt att bedriva utveckling användarcentrerat är prototyping. Flera författare, exempelvis Sommerville (2001) beskriver det teoretiska ramverket för prototyping. INUSE [2] beskriver User-Centred design (UCd) på ett kortfattat och relativt handfast sätt och som en prototypingmetod att uppnå användarvänlig programvara. Man för fram 4 grundläggande karaktäristika hos UCd vilka är i linje med ISO 13407 (1998):

1. an appropriate allocation of function between user and system
2. the active involvement of users
3. iteration of design solutions
4. multi-disciplinary design teams

Som vi kan utläsa syftar användarcentrerad systemutveckling till att göra system användbara. Användbarhet (usability) definieras i ISO 9241-11 (1998) som:

> *”the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”.*

INUSE [2] konkretiserar de steg som Bevan för fram kring systemutveckling och som ovan redogjorts för i figur 14 och med det uttalade målet att skapa applikationer som uppfyller de ovan formulerade kriterierna för ”usability”.

En reservation i sammanhanget av såväl Bevans artikel som INUSE [2] är de olika förutsättningarna som gäller målgruppen. Dessa riktar mer eller mindre in sig på utveckling inom organisationer och med mer eller mindre kända användare. Den egentliga målgrupp för resultatet av detta arbete är inte känd utan resultatet av den systemutveckling som inleds är att betrakta som det Sommerville (2001) benämner ”Generic products” dvs en produkt som är producerad för att säljas på en öppen marknad till envar som är villig att köpa den. Motsatsen, dvs det som Bevan (1999) och INUSE [2] är mera inne på ”Customized products” är utvecklat speciellt för en kund.

Andersen (1994) för fram en liknande modell när han beskriver prototyping. Även han har ett ganska uttalat fokus på systemutveckling i en känd grupp. Den största skillnaden mellan Andersen (1994) och INUSE [2] ligger snarast i Andersens något mindre framskjutna fokusering på verksamheten.

> *"Men verksamhetsstudien är i prototyping upplagd på ett annat sätt än i andra modeller. Den skall bara ge ett underlag till framställning av den första prototypen".*
>
> (Andersen, 1994, s. 411)

En helt annan fokusering på verksamheten hittar vi hos Greenbaum & Kyng (1991) som även de arbetar med prototyper, mock-ups, storyboards et cetera men som mycket starkt betonar vikten av att förstå både människor och sammanhang vari man utvecklar systemet.

Sommerville (2001) gör åtskillnad mellan å ena sidan Evolutionary prototyping och å andra sidan Throw-away prototyping. Vid evolutionär prototyputveckling är målet att själva prototypen gradvis skall förfinas och så småningom bli själva applikationen. Vid throw-away är målet att få fram en kravspecifikation för systemutveckling, dessa senare tankar sammanfaller med Andersens syn på arbetet. Sommerville (2001) går till och med så långt att han varnar för frestelsen att realisera prototypen till färdig applikation. Vidare har han inte alls samma fokus på miljön i vilken man arbetar med prototyping även om han inte uttrycker explicit att prototyping är för organisationer med kända användare så uttrycker han inte heller motsatsen – att prototyping är olämpligt för generisk mjukvaruutveckling.

Avslutningsvis kan vi då konstatera att prototyping är en lämplig modell för att ta fram de systemförslag som skall utvärderas. Metoden har goda förutsättningar att komma så nära idealen kring Quality in Use som möjligt. Samtidigt som resultatet av prototypingen är ett konkret resultat som är relativt lätt att utvärdera.

2.5 Kravprocessen

Att hantera kravprocessen är ett sätt att fånga de tänkta användarnas krav på vår mellanliggande programvara. Att hantera processen är en viktig del av arbetet med utvärderingen av de prototyper som vi utvecklar. Det är därför av vikt att ha kunskap om vilka typer av krav man kan förvänta sig att identifiera respektive inte upptäcka beroende på metod.

Kravprocessen är en process som är en del i en större process, mjukvaruutveckling, som enligt Bray (2002) kan delas in i fem olika steg: insamling, analys, specificering, design av användargränssnitt och validering. Insamling går ut på att samla in information som ska ligga till grund för kravprocessen och exempelvis berörs frågor som; vad som bör samlas in, vad som kan samlas in och hur detta låter sig göras. Analys av befintligt system avser att undersöka och dokumentera problemområdet för att skapa förståelse. Vidare kan specificeringen beskrivas som processen att skapa och definiera egenskaperna hos det nya systemet som kommer att resultera i de eftersträvade effekterna i problemområdet. Design av användargränssnitt är en process som kräver expertkunskap och ofta genererar komplexa beskrivningar hos systemet. Valideringen syftar till att

undvika missförstånd som kan uppstå under tiden kravprocessen pågår. Det är därför viktigt att kontinuerligt validera kravprocessen för att säkerställa rätt funktionalitet hos det nya systemet. Valideringen bör ske genom att testa och kontrollera på många olika nivåer, t ex genom att låta den som intervjuats läsa igenom intervjuresultat (Bray, 2002).

Sommerville (2001) förklarar begreppen krav (requirements) och kravprocessen som funktioner och villkor respektive processerna att hitta, analysera, dokumentera och validera dessa krav vilket stämmer bra med vad Bray (2002) säger. Vidare lyfter Sommerville (2001) fram att det finns olika abstraktionsnivåer när det gäller specificering av krav. Användarkrav är en specificering med relativt låg detaljeringsgrad medan systemkrav är mycket mer detaljerade.

Vidare finns det enligt Sommerville (2001) funktionella och icke funktionella krav samt domänkrav. Funktionella krav är de krav som måste vara uppfyllda för att funktionaliteten i systemet ska kunna fungera. De icke-funktionella kraven är snarare villkor för funktionerna som exempelvis skulle kunna vara tidsaspekter, standards, etc. Här skiljer sig Brays (2002) benämning en aning eftersom han kallar icke-funktionella krav för prestandakrav istället, innehållet beskrivs dock likadant. Vidare beskriver Sommerville (2001) domänkrav som härrör från applikationens domän och den påverkan domänen har på applikationen. Han påtalar också vikten av att kraven bör specificeras så att användarna verkligen förstår dem. Ofta resulterar detta i beskrivningar som specificeras med hjälp av vanlig text vilket enligt Sommerville (2001) kan innebära problem med oklarheter, förvirring och krav som har klumpas ihop.

Sommerville (2001) talar för att kravprocessen helt enkelt bör fokuseras på att fånga de huvudsakliga kraven. Han menar att risken annars är överhängande att användarna inte förstår beskrivningarna på grund av de är för detaljerade.

Enligt Bray (2002) är det viktigt att skilja på intern och extern design av det skälet att extern design hör till kravprocessen medan intern design mer är fråga om själva designfasen i systemutvecklingsprocessen. Bray (2002) lyfter fram detta genom att beskriva ett exempel där han antar att vi vill ha en mänsklig robot. Externa designen berör till exempel att den måste se ut och röra sig som en människa medan den interna designen snarare berör hur den ska kunna fungera (Bray, 2002). Detta stämmer mycket väl överens med de underliggande teorierna i QIU som Bevan (1999) beskriver.

När det gäller insamling av material under kravprocessen förespråkar Hawryszkiewycz (2001) fyra huvudsakliga sätt; att ställa frågor, att observera, använda prototyping och formella sessioner. Att ställa frågor innebär enligt Hawryszkiewycz (2001) t ex intervjuer och formulär medan observationer inkluderar etnografi och eller deltagande observation. Vidare menar han att prototyping kan indelas i två olika syften, dels för att fånga upp krav och dels för att förbättra ett gränssnitt.

Maciaszek (2001) delar istället in insamling av material i traditionella respektive moderna metoder. Till de traditionella hör intervjuer av kunder och domänexperter, frågeformulär, observationer samt studier av dokument och system. Till de moderna hör prototyping, joint application development (JAD) samt rapid application development (RAD). Maciaszek (2001) beskriver prototyping som den mest frekvent använda metoden för att visualisera och få feedback på ett system. Han säger vidare att en prototyp är ett väldigt effektivt sätt att hitta krav som är svåra att få fram på andra sätt ifrån exempelvis en kund.
Både Hawryszkiewycz (2001) och Maciaszek (2001) talar om intervjuer som det primära sättet att samla in information. Hawryszkiewycz (2001) tycker det är viktigt att lyfta fram problem och prioriteringar för att diskutera det resultat som kommit fram av processen. Maciaszek (2001) nämner vidare att de flesta intervjuer görs med kunder, men trycker på att intervjuer med domänexperter är ett enkelt sätt att överföra domänkunskap till systemutvecklarna. Kunderna har ofta enligt Maciaszek (2001) bara en vag bild av sina krav, vilket gör det intressant att intervjua s k domänexperter. Det är ett resonemang som stämmer mycket väl med det arbete som utförs i denna studie, vi har valt att låta intervjua några av Sveriges relativt få domänexperter inom RFID-området.

En viktig aspekt i all systemutveckling vilket även innefattar kravprocessen är att processkvalitet är något som har ett starkt samband med produktkvalitet (Sommerville, 2001; Håkansson, 2000). Sommerville (2001, s. 543) menar att *"An underlying assumption of quality management is that the quality of the developement process directly affects the quality of delivered products"*.

Avslutningsvis konstaterar vi att prototyping, om den används rätt, kan vara värdefull när det gäller att förstå de krav som blivande eller tänkta användare kan komma att rikta mot programvaran. Vi finner också stöd för vår tanke om att utnyttja prototyper som diskussionsunderlag i samband med intervjuerna av domänexperterna i utvecklingsarbetet. Domänexperterna framstår också som en mycket värdefull informationskälla eftersom kunden ofta har en väldigt vag bild av vilka krav som bör ställas.

2.6 Småskaliga tillämpningar

Tidigare har begreppet småskaliga tillämpningar definierats som utvecklad med relativt knappa medel för användning i relativt begränsad omfattning, se avsnitt 1.1 begreppsdefinitioner. En sådan definition är en förenkling, allt är utvecklat med begränsade resurser och i begränsad upplaga. Heiat & Heiat (1997) för i en artikel fram tankar kring småskaliga applikationer som närmar sig en definition av begreppet. Artikeln tar upp frågan kring hur man beräknar arbetsinsats (effort) när man utvecklar småskaliga applikationer, bland projekt man undersökt har man valt ut applikationer som uppfyller ett antal kriterier varav några är relaterade till storleken på projekten:

> a. *They were small. The projects' size ranged from 325 to 2940 lines of code.*
> b. *They were all written in 4GL database programming language.*
> c. *They were developed by a small team of one to three systems analyst and programmers. Several users were involved with each project.*
>
> (Heiat & Heiat, 1997, s. 10)

Laitinen (2000) fyller på med ytterligare tankar som kan bidra till att nå en hållbar definition. Enligt honom är skillnaden på ett stort och ett litet projekt något annat än en skillnad som kan åskådliggöras med siffror.

> *"...a person might do a job differently precisely because the job's reporting and coordination components are much smaller. This is not just a change in volume; it is a change in kind".*
>
> (Laitinen, 2000, s. 80)

Boddie (2000) menar att en skillnad mellan små och stora projekt gör sig gällande i ledningen av dem och då avseende kommunikation och integration inom projekten. Behovet av kommunikation och integration ökar med avseende på projektets storlek och därmed förändras arbetssättet inom projektet som ett resultat av projektets storlek.

Svensson (2004) kompletterar med att även tiden är en faktor som måste vägas in. Liksom att den önskade funktionaliteten sannolikt skiljer sig åt mellan små- och storskaliga tillämpningar.

En mera elaborerad definition av begreppet småskalig för den här uppsatsen efter vår tolkning lyder:

- Tillämpningen är av den storleken att den på rimlig tid kunnat utvecklas av 1-3 systemerare och programmerare, förmodligen mera generalister än i stora projekt.
- Tillämpningen är skapad i en grupp där behovet av formella tekniker för kommunikation och integration inom projektet är begränsade då alla i projektgruppen kan ha direktkontakt.
- Viktigare än hur omfattande själva programvaran är mätt i antal rader kod är med vilket förhållningssätt (kind) sätt den är tillkommen. Återanvändning av kod, färdiga moduler mm gör att rader av kod inte säger så mycket om den insats som krävs för realisering.
- Användarna inte tillhör de mest krävande i termer av vilka specialiserade funktioner man önskar av systemet.
- Antalet entiteter (taggar) i systemet är ingen funktion av tillämpningens storlek och därmed oväsentligt.
- Värdet av den information som systemet genererar motiverar enbart en mindre tillämpning.

3 Metod

Kapitlet beskriver hur vi vill uppnå målen med vårt arbete – vår metod. Metod är den mer hantverksmässiga sidan av den vetenskapliga verksamheten. Men metod är något mer än bara undersökningstekniker. Vi inleder med att diskutera metod på ett mera generellt plan, precisera vårt metodval något och argumentera för det valet, i slutet av kapitlet omsätts metodvalet i en mer konkret beskrivning av arbetets design.

3.1 Val av metod

Forskaren måste behärska olika sätt hur data samlas in, organiseras, bearbetas, analyseras och tolkas så att man uppfyller kraven för kontrollerbarhet. Redskapen för att lösa problem och för att nå ny kunskap är metod (Holme & Krohn-Solvang, 2001). Halvorsen (1992) menar att det dock är viktigt att komma ihåg att metoden skall utgöra ett hjälpmedel och inte utgöra ett mål i sig själv.

En vanlig uppfattning i metodböcker är att frågan eller problemet styr val av metod, Holme & Krohn Solvang (2001) skriver exempelvis:

> *"Valet av metod skall ske utifrån den problemformulering vi gjort för vår undersökning".*
>
> (s. 85)

Problemformuleringen skall således styra valet av metod. Holme & Krohn Solvang (2001) utgår från att man gör ett val av metod och att detta val grundas på det problem man formulerat;

Hur kan en generell, användarvänlig programvara utformas för att koppla samman RFID-identiteter med en databas, och därigenom underlätta skapandet av småskaliga RFID-baserade informationssystem?

Vi skall således undersöka hur något *kan utformas*. Vår inledande tanke kring metoden är att genom ett systemutvecklingsprojekt ta fram förslag på utformning av den mellanliggande programvaran. Förslaget utvärderas sedan genom intervjuer – en kvalitativ studie. Starrin et al (1991) menar att huvudargumenten för den kvalitativa metoden är förståelse, intersubjektivitet, teorigenerering, validitet, upptäckt, variation och framför allt nyfikenhet.

Metodvalet har vi sedan diskuterat utifrån ett antal kriterier som Holme & Krohn Solvang (2001) ställer upp som utmärkande drag för kvalitativ och kvantitativ metod.

Tabell 4. Utmärkande drag för kvalitativ och kvantitativ metod. Förenklad efter Holme & Krohn Solvang (2001).

	Kvantitativa metoder	Kvalitativa metoder
1	Avspegling av kvantitativ variation, precision	Återgivning av kvalitativ variation, flexibilitet
2	Går på bredden	Går på djupet
3	Systematiska och strukturerade observationer	Osystematiska och ostrukturerade observationer
4	Intresse på det gemensamma	Intresse för det säregna
5	Avstånd till det undersökta	Närhet till det undersökta
6	Intresse för åtskilda variabler	Intresse för sammanhang och strukturer
7	Beskrivning och förklaring	Beskrivning och förståelse
8	Åskådare eller manipulatör	Deltagare eller aktör
9	Jag-det-relation till den undersökte	Jag-du-relation till den undersökte

3.2 Argument för vald metod

1. Kvantitativ eller kvalitativ variation

RFID-tekniken är i sin nuvarande tappning relativt ny, de flesta tillämpningar vi ser är storskaliga lösningar. Med uppsatsens fokus på relativ småskalighet är det därför tveksamt med någon typ av kvantitativ undersökning med inriktning på kvantitativ variation. Vi vet dessutom lite om de tänkta användarna, och vilka de är. Det finns med andra ord inte så mycket att ”mäta” på, snarare är intresset att förstå, att ge en tolkning, en bild av hur en generell programvara kan se ut. Vårt fokus ligger på att beskriva och i någon mån utveckla en tänkt mellanliggande programvara. Utvärderingen kommer att ske genom att värdera utsagor från några initierade inom området, resultatet ligger till grund för fortsatt utveckling av den tänkta programvaran.

2. Bredd eller djup

Kvantitativa metoder har fokus på bredd i undersökningen, typiskt samlar man relativt lite information om relativt många undersökningsobjekt. För kvalitativa undersökningar kan det omvända sägas gälla, här samlar vi mycket information om ett mindre antal undersökningsobjekt och vi är intresserade av helhet. Undersökarens egen referensram utgör ingen begränsning i respondentens möjligheter att uttrycka sig. Antalet respondenter om vi håller oss inom Sverige får bedömas som begränsat. Resultatet av det här arbetet, en tänkt programvara kan dock i sin tur bli föremål för en kvantitativ studie.

3. Systematiskt och strukturerat eller inte

Kvantitativa undersökningar karaktäriseras av strukturering, vi arbetar med samma frågor och samma svarsalternativ och forskaren styr insamlingen av information. I kvalitativa undersökningar är flexibilitet ett nyckelord, frågor och rent av arbetssätt kan omformuleras under pågående arbete. Vi kommer att diskutera med våra undersökningspersoner, fråga om just deras erfarenheter, vad de tror. Säkert kommer följdfrågor och oväntade vändningar. Genom att arbeta kvalitativt kan arbetet anpassas att följa nya förutsättningar.

4. Intresse för det gemensamma eller säregna

När vi arbetar kvantitativt vill vi ofta finna möjligheter att generalisera, dvs vi vill från ett urval undersökningsobjekt dra slutsatser om en population, vi vill uttala oss om det som är gemensamt för gruppen. I en kvalitativ undersökning har vi som regel ingen ambition eller möjlighet att generalisera våra resultat. Vi antar att frågor om småskaliga RFID-tillämpningar sysselsätter ett antal människor. Här har vi dock den här studiens komplikation, vi vill peka på en modell för hur ”man kan göra”. ”Verkligheten” erbjuder dock ingen eller mycket begränsad möjlighet att skaffa oss generaliserbar kunskap. Alternativet som vi ser det är att vi erbjuder en aspekt (Asplund, 1970) av denna verklighet och att vi är tydliga med att detta inte är det enda och sanna ”svaret”.

5. Avstånd eller närhet till det undersökta

I en kvantitativ undersökning har vi en syn på den vi undersöker som en i populationen, vem han eller hon är och hennes speciella historia utgör inte det primära intresset för oss. Vi behöver inte träffa henne, vi har ett avstånd till den vi undersöker. När vi arbetar kvalitativt gäller det omvända, personen eller företeelsen vi just nu undersöker är den som är det viktiga och hans eller hennes historia är det allt kretsar kring, vi strävar efter närhet. vi bedömer att närhet till det vi undersöker är en absolut förutsättning i den här typen av studie.

6. Intresse för variabler eller sammanhang

I en kvantitativ undersökning är vi ofta intresserade av att finna kausalsamband, dvs. en faktor eller variabel som styr en annan. I den kvalitativa undersökningen utgår vi ofta från att världen är så komplex att olika samband, om de finns, är utomordentligt svåra att finna. Vår bedömning här är att frågor om kausalsamband inte är relevanta. Frågorna är så komplexa att samband om de finns ställer sig extremt svåra att finna. Vi fokuserar vår undersökning på helhet snarare än delar.

7. Beskrivning och förklaring eller förståelse

Den kvantitativa undersökningen syftar typiskt till att beskriva och förklara något, medan den kvalitativa syftar till att beskriva och förstå något. Här handlar valet förutom att i båda fallen att beskriva även om ett val mellan det objektiva – sanna och det subjektiva – tolkade. Ett ställningstagande för oss är att det inte finns en objektiv sanning i detta, ibland är en lösning bra i nästa sammanhang är samma lösning trots påfallande likheter i situationen dålig. Allt är på så sätt mer eller mindre bra beroende på kontexten. Vårt resultat blir vår tolkning av verkligheten så som den beskrivs av intervjupersonerna.

8. Åskådare eller deltagare

Den kvantitativt arbetande forskaren är en åskådare. Han eller hon har ingen, och skall inte ha någon ambition att gripa in i undersökningssituationen, forskaren är oberoende och värderar inte situationen. Däremot, och detta är viktigt, innebär inte detta att forskaren är utan engagemang. Forskningen kan mycket väl syfta till att åstadkomma en förändring, men den utförs inte i undersökningen. Detta ställningstagande är egentligen redan gjort när frågan formu-

lerats, vi tror att den här tekniken kan användas på ett i grunden positivt sätt. Implicit innebär detta att vi som undersökare utgår ifrån att vi har något att tillföra dem vi intervjuar och att detta nästintill är en förutsättning att få tillgång till de miljöer där undersökningen skall utföras.

9. Jag-det eller jag-du

Den kvantitative forskaren har som redan antytts en relation till den han/hon undersöker som inte är personlig. Den kvalitative forskaren lär känna sina undersökningspersoner, det är själva poängen med arbetet. Som tidigare skrivits anser vi att en förutsättning för att överhuvudtaget kunna komma till tals med någon som kan uttrycka några kvalificerade uppfattningar kring problemet är att vi kan skapa en personlig relation.

3.3 Ett explorativt förhållningssätt

Redan tidigt har vi konstaterat att den exakta problemformuleringen varit svår att precisera, vi har inte funnit någon tidigare forskning som ligger nära det vi gör och tekniken vi arbetar med är i en utveckling som går mycket snabbt.

Emory & Cooper (1991) beskriver design av forskningsprocessen som en komplex fråga och den kan betraktas från minst 8 olika perspektiv. Det första perspektivet är:

> *"The degree to which the research problem has been crystallized the study may be either exploratory or formal".*
>
> (Emory & Cooper, 1991, s. 139)

Vi går här inte in på de följande perspektiven utan nöjer oss med att konstatera att sättet att uttrycka sig skiljer sig från Holme & Solvang. Och att just den här punkten är viktig för vårt arbete. Emory & Cooper skiljer som vi sett mellan:

- **Exploratory studies** tend to be loosely structured with an objective of learning what the major research task are to be.
- The **formal study** begins where the exploration leaves off - it begins with a hypothesis or question and involves precise procedures and data source specifications.

(Emory & Cooper, 1991, s. 140)

Emory & Cooper (1991) konstaterar att denna tudelning är mindre precis än andra klassificeringar, alla studier har mer eller mindre drag av båda riktningarna inom sig. Vad vi upplever i vårt arbete med en teknik som är ganska ny, med en målgrupp som är ny, med ett fokus på användning som avviker från gängse et cetera, är att vi måste hålla dörren öppen för mer explorativa drag i vårt arbete. Undersökningen kommer därför även att ha drag av ett formulativt, explorativt eller problemsökande arbetssätt (Carlsson, 1990). Det här kan ta sig konsekvenser som att teoridelen under arbetets gång kan behöva justeras såväl i inriktning som nivå.

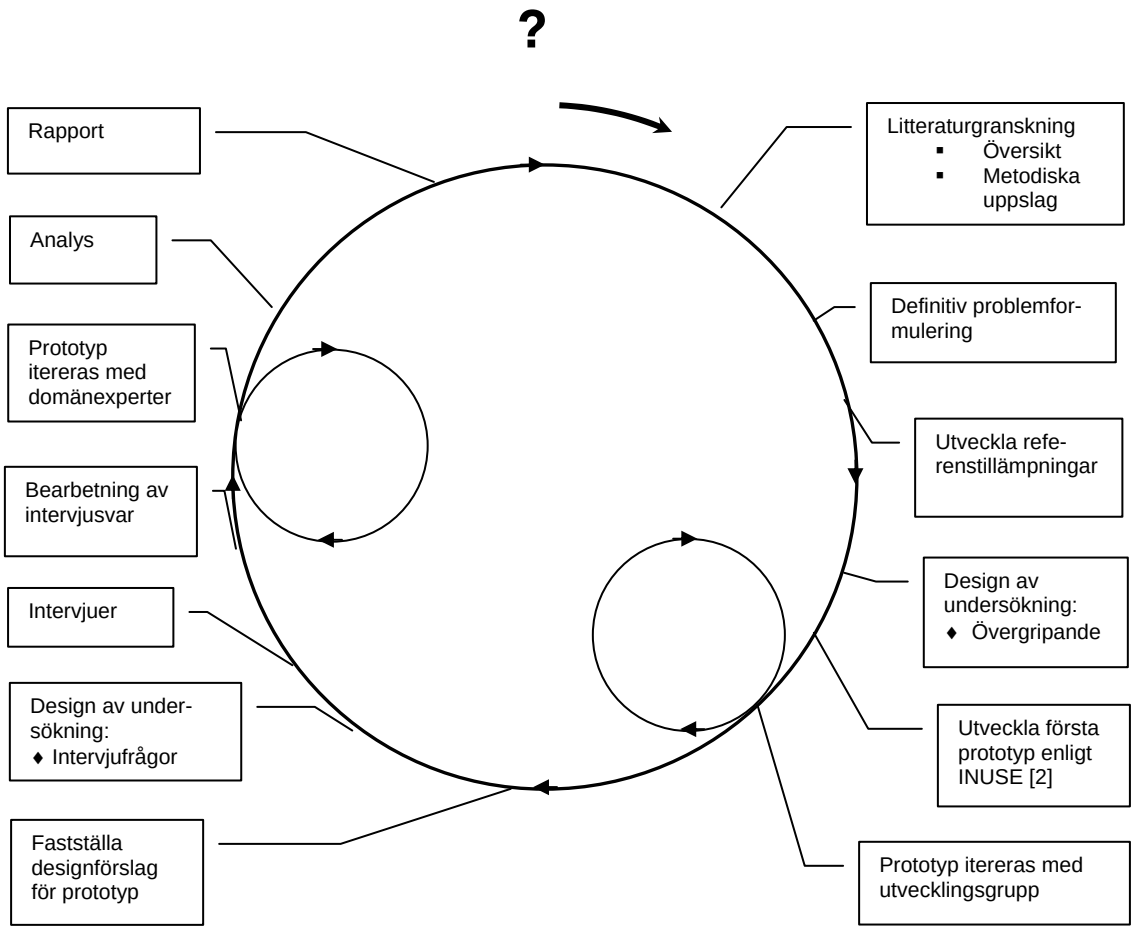
3.4 Design – vald metod

För att få svar på vår frågeställning kommer vi att utarbeta en prototyp som ett antal grafiska gränssnitt med begränsad funktionalitet. Utvecklingen av förslagen kommer att ske enligt den modell som INUSE [2] beskriver samt standarden ISO 13407 ”Human Centred Design Process for Interactive Systems” som den beskrivs i INUSE [2]. Dessa prototyper kommer sedan att utvärderas med en grupp domänexperter efter modellen om Quality in Use (Bevan, 1999). De första prototyperna kommer vi att utvärdera kring mer praktiska frågeställningar tillsammans med personer vi har runt omkring oss och som har god vana inom olika relevanta områden. Den här gruppen som vi kallar utvecklingsgruppen, består av studenter som nyligen slutfört sin magisterutbildning samt kollegor som har goda kunskaper inom relevanta områden. Tanken med att engagera den tidiga utvecklingsgruppen innan den egentliga utvärderingen tillsammans med domänexperterna är att få en så mogen första prototyp som möjligt. Användarcentrerad systemutveckling är enligt INUSE [2] en process som bygger på samarbete, där det är viktigt att knyta till sig många slag av olika kompetenser.

Prototypen skall tillsammans med våra förklaringar ge en känsla för hur en generell programvara för småskaliga RFID-tillämpningar skulle kunna fungera. Den mera mogna prototypen kommer att presenteras för några människor som arbetar professionellt med RFID-tillämpningar och personerna kommer att intervjuas kring frågor om utformning av en sådan programvara. De tidiga itereringarna kommer också att ligga till grund för klarare preciseringar av frågor till domänexperterna. Preciseringarna framgår av systemutvecklingsdokumentationen i bilaga 1 samt under redovisning av prototypen i avsnitt 4.1 och 4.2.

3.5 Gången i arbetet

För att beakta de metodmässiga och systemutvecklingsmässiga aspekter som förts fram, har vi valt att sammanställa arbetsgången i följande bild:



Figur 15. Schematisk bild av gången i uppsatsarbetet.

3.6 Insamling, bearbetning och analys av data

3.6.1 Litteratur

Teoridelen av vårt arbete är i huvudsak hämtad från ett antal artiklar inom relevanta områden, artiklarna har sökts i olika databaser som ACM, IEEE och Science Direct. Även vår handledare har gett tips om relevanta artiklar. Vi har läst ett antal böcker som allmänt betraktas som centrala inom sina respektive områden, exempelvis Date (2004), Finkenzeller (1999) och Sommerville (2001).

En svårighet vi upplevt är att vårt uppsatsområde inte synes vara undersökt. Det finns många artiklar skrivna om delarna som uppsatsen fokuserar på, men ser man till artiklar som berör två eller flera av delarna så är innehållet i dessa inte relevant i sammanhanget. En annan intressant iakttagelse är att antalet nya artiklar kring RFID ökat starkt det senaste året.

En sökning i IEEE:s och Science Direct:s databaser den 13 april 2004 gav enligt tabell 5 följande antal träffar:

Tabell 5. Antal träffar på sökord I IEEE:s/Science Direct:s artikeldatabaser 2004-04-13.

Sökord	IEEE	Science Direct
RFID	101	14
Radio frequency identification	66	11
RFID and application	54	3
Radio frequency identification and application	34	3
RFID and database	3	2
Radio frequency identification and database	1	0
RFID and functionality	2	1
Radio frequency identification and functionality	1	1
Sökmönster	All fields; year 1995-2004, Alla publikations-typer	Search in abstract, title, keyword. Year 1995-2003. All sci-ences.

Tabell 5 är inte någon fullständig redogörelse för de litteratursökningar vi gjort utan speglar ett förhållande när det gäller vår valda forskningsfråga. Området RFID och sökord som är relevanta för den här studien är som vi kan se det inte beforskat. De artiklar vi valt ut innehåller aspekter på de frågor som är intressanta för arbetets vidkommande. I övrigt har vi arbetat med inom forskning välkänd litteratur.

3.6.2 Intervjuundersökningen

> *"I den kvalitativa forskningsintervjun bygger man upp kunskap: det rör sig bokstavligen om ett samspel, om ett utbyte av syn-punkter mellan två personer som samtalar om ett ämne av gemensamt intresse."*
>
> (Kvale, 1997, s. 21)

För att få ökad klarhet kring vilken funktionalitet en programvara som den vi beskriver bör ha kommer vi att genomföra ett antal intervjuer. Intervjuerna kommer att genomföras med personer som har goda kunskaper kring RFID och applikationer för att använda tekniken. Vi utgår från Kvale (1997) när vi utformar intervjudelen i vårt arbete, som föreslår 7 stadier i en intervjuundersökning:

1. Tematisering
Består av den teoretiska analys av det tema som skall undersökas (Kvale, 1997). Nyckelfrågorna enligt Kvale är:

- Vad: att skaffa sig förkunskaper om det ämne som skall undersökas.
- Varför: att klargöra syftet med undersökningen.
- Hur: att förvärva kunskap om olika intervju och analystekniker.

I den här undersökningen är tematiseringens vad och varför klara genom formulering av problem och syfte, teorigenomgång samt utveckling av referenstillämpningar, prototyper och metodfrågor.

2. Planering
Här planeras undersökningens 7 stadier med hänsyn till vilken kunskap som eftersträvas. Vilken kunskap vi eftersträvar framgår av arbetets problemformulering, samt utvecklade resonemang ovan.

3. Intervju
Intervjuerna genomförs enligt en utarbetad intervjuguide grundad på innehållet i Quality in Use och iteration av prototyper. Intervjuguiden återfinns som bilaga 3. De personer som varit intressanta att intervjua har först kontaktats per telefon eller personligt samtal. Alla vi frågat har tackat ja. Intervjuerna har skett på respektive respondents arbetsplats. Intervjuguiden har behandlat de olika områden som vi varit intresserade av att få veta mer om och vi har specificerat frågor inom respektive område. Frågorna som har ställts upp i förväg är de frågor vi har velat få svar på från samtliga respondenter. Till samtliga respondenter ställts ytterligare frågor samt att vi ibland även varit tvungna att formulera om frågeställningarna under intervjuerna. Vissa frågor har behövt förtydligas och prototypen har behövt förklaras på olika sätt. Intervjuerna har således haft drag av att vara både strukturerade och ostrukturerade. Den sammanlagda tiden för de tre intervjuerna uppgår till cirka 4 timmar. Varje intervju har varat i minst en timma.

4. Utskrift
Under intervjuerna användes en Mp3-spelare för att lättare kunna föra en naturlig dialog samt för att inte få faktafel. Genom att använda Mp3-spelare vid intervjuerna missar man heller inte någon information från respondenterna. I och för sig garanterar detta inte att informationen är den korrekta. Som intervjuare kan man tolka informationen fel likaväl som respondenterna kan uppge felaktiga uppgifter. Möjligheten till att kontrollera korrektheten i den information som intervjupersonerna ger är mycket begränsad.

Samtliga intervjuer har skrivits ut enligt exakt ordalydelse, vilket gått bra eftersom vi använt oss av bandspelare. Arbetet med dessa utskrifter har tagit cirka tre till fyra gånger så lång tid som själva intervjutiden. Detta underlag biläggs inte uppsatsen utan används endast som underlag till resultatpresentationen. Att inte bilägga detta gör vi just med hänsyn till den anonymisering vi valt att ha(se vidare avsnitt 3.6.5 Anonymisering).

5. Analys
I en kvalitativ studie, där intervjuer genomförs, erhåller man en mängd data och därmed text (intervjuutskrifterna). Att ta vara på detta innehåll görs lämpligen genom en textanalys. Enligt Tebelius, (1987), behandlas datamaterialet på två nivåer:

1) Datasammanfattning genomförs för att göra texten hanterbar. Allt material via intervjuerna grupperas och kategoriseras efter sammanhang. Därefter gör man en subjektiv kodning vilket innebär att meningsbärande enheter markeras.

Utifrån denna datasammanfattning ställer man samman resultatet, den empiriska referensramen.
2) I analysen av studien kopplas sedan den teoretiska och empiriska referensramen ihop med avseende på att se mönster och sammanhang relevanta för problemområdet och uppställda syften.

Utifrån teorin, har vi bl a valt Bevans modell, (1999), Quality in use för att kategorisera vårt material med utgångspunkt från de olika dimensioner och faktorer som framkommer i textanalysen. För ändamålet har vi gjort en analysmall som kategoriserar frågorna, denna återfinns i bilaga 5.

6. Verifiering
Intervjuresultatets generaliserbarhet, reliabilitet och validitet fastställs. Se 3.6.8

7. Rapportering
Resultatet rapporteras.

3.6.3 Val av intervjupersoner

Tre personer har intervjuats, alla arbetar på företag med verksamhet inom RFID-området. Kontakt och kännedom om de här personerna har vi fått genom våra respektive arbeten. Inom RFID-området har vi vetskap om ytterligare en handfull personer i landet som varit intressanta att intervjua. Praktiska och ekonomiska avväganden har gjort att vi begränsat oss till dem som finns i vår region (ca 15 mil runt Halmstad).

De personer vi valt ut och intervjuat är:

Tabell 6. Intervjupersoner

Respondent	Företag	URL
Bengt Sunesson	IdentCode AB	www.identcode.se
Mats Jirander	Est AB	www.est.se
Per-Arne Wiberg	Free2move AB	www.free2move.se

3.6.4 Urval -urvalsdiskussion

Intervjuperson 1 (numreringen har inget samband med ordningsföljden i tabell 6) har djup kunskap och erfarenhet kring RFID-tillämpningar, bland annat ansvarar hans företag för tekniken kring tidtagning i Vasaloppet och har varit med och byggt upp det systemet. Vidare har han stor kunskap när det gäller att utveckla RFID-tillämpningar. Intervjuperson 2 har mycket lång erfarenhet inom RFID-området, förutom teknisk bakgrund har han kunskaper även inom bland annat juridik och ekonomi. Företaget han arbetar med är specialiserade inom datafångst grundat på flera olika tekniksslag utöver RFID. Intervjuperson 3 har mångårig och bred kunskap inom området trådlös kommunikation, innovations- och utvecklingsarbete samt utvecklar egen hårdvara och system. Han är vidare publicerad i flera vetenskapliga sammanhang.

Intervjupersonerna har olika inriktning inom området, de har kunskaper inom såväl, hård- som programvara och system. Det finns internationell erfarenhet i

gruppen. Det finns bland intervjupersonerna kunskaper inom forskning, såväl egen som handledning av doktorander och/eller studenter på magister-/civilingenjörsnivå. Naturligtvis även lång industriell praktisk erfarenhet som företagare, konsulter och anställda i teknikföretag. Vår bedömning är vi visavi arbetets syfte har ett tillräckligt underlag genom de intervjuer vi gjort.

3.6.5 Anonymisering och sekretess

Anonymisering är en viktig fråga när det gäller redovisning av resultatet från intervjuer. Många gånger kan läsaren uppfatta det som mer intressant, och mer verkligt, om intervjuerna återges öppet och med tydlig hänvisning till vem som uttryckt sig. Å andra sidan måste respondenternas integritet beaktas. De tre intervjupersonerna har samtliga medgivit att namn och företagsuppgifter får publiceras, dock vill man ha respektive utsaga anonym. För den här studien är det inte heller av större betydelse vem som säger vad, det är den samlade kunskapen och erfarenheten som är betydelsefull. I något fall kan respektive respondents utsaga härledas till en person. Respondenterna är vidtalade och medvetna om detta, alla tre har medgivit publiceringen på det sätt som nu sker.

Vid samtliga intervjuer har vi bett respondenterna tala om för oss om någon information är känslig/hemlig och att det av den anledningen är olämpligt att återge i resultatredovisningen. Endast vid en av intervjuerna har känslig information delgetts oss. Denna information har dock inte varit av betydelse för undersökningen och studiens syften, utan rörde ett pågående utvecklingsprojekt.

3.6.6 Prototypframtagning

Som vi tidigare sett i teorigenomgången finns det enligt INUSE [2] och ISO 13407 fyra principer när man utvecklar programvara enligt en användarcentrerad modell. Syftet med dessa fyra principer är att inkorporera användarens perspektiv i utvecklingsarbetet (INUSE [2], 1.3). När vi sedan rör oss från det mer abstrakta mot det mer konkreta finns det enligt ISO 13407 fyra steg som skall gås igenom för att få med usability-kraven i utvecklingsarbetet. Under flera av punkterna finns ytterligare underrubriker.

- Understand and specify the context of use.
- Specify the user and organizational requirements.
- Produce designs and prototypes.
- Carry out user-based assessment.

Arbetet att ta fram ett systemförslag kommer att följa dessa fyra punkter, och som redovisats tidigare de fyra grundläggande karaktäristika hos UCd/ ISO 13407 (1998):

1. an appropriate allocation of function between user and system.
2. the active involvement of users.
3. iteration of design solutions.
4. multi-disciplinary design teams.

Prototyperna presenteras i avsnitt 4.1 och 4.2. Upprättad dokumentation av prototypframtagning redovisas som bilaga 2.

3.6.7 Problem under arbetets gång

En fråga som vållade vissa bekymmer var hur vi skulle kunna hitta slutanvändarna för den mellanliggande programvaran. Förvisso finns slutanvändare för RFID-tekniken men som vi konstaterat tillhör de flesta gruppen storskaliga användare. Återstår således presumtiva slutanvändare men av naturliga skäl vet ingen vilka de är och många av dem känner inte heller till tekniken. Detta problem framgick i samband med intervjuerna., se vidare avsnitt 4.3.3 och 4.3.12. Vidare ställde detta problem till det i samband med att prototyping som metod hela tiden utgår ifrån att slutanvändarna är kända. Vi upplevde att det inte alltid är så verkligheten fungerar, användarna är inte alltid kända. Ottosson (1993, s. 11) menar att *"marknaden har ett latent behov av produkter som inte redan finns att tillgå. Marknaden måste alltid läras att få ett behov av en ny produkt"*. I undersökningen fick vi därför vända oss till olika experter för att på så vis indirekt fånga slutanvändarnas behov och krav. Som framgår av avsnitt 2.5 kravprocessen är det även en styrka i samband med kravprocessen att vända sig just till s k domänexperter. Effekterna av problemet minskade successivt, då vår prototyp under arbetets gång och de intervjuer vi genomförde, kom att övergå från att bli en branschspecifik lösning till att bli en helt generell mellanliggande programvara. Vi bedömer att det som inledningsvis uppfattades som ett problem snarare påverkat arbetet positivt snarare än negativt.

Vårt arbete knyter ihop olika ämnesområden, vilket stämmer bra överens med informatikämnets natur. Informatik är av samhällsvetenskaplig inriktning, vilket våra respondenter inte varit, utan de företräder mer det rent tekniska området. Detta har gjort att de begrepp som använts av oss och respondenterna försvårat kommunikationen en aning jämfört med om vi alla hade tillhört samma ämnesområde. Mycket tid har behövts för att säkerställa att vi har pratat om samma saker och för att förklara vad som i olika situationer avses. Förutom en del missförstånd i samband med intervjuerna har detta endast påverkat att det tagit längre tid att genomföra intervjuerna. Tiden har dock inte varit ett problem eftersom vi hade avsatt tillräckligt med tid för intervjuerna. Men att vi inte har haft samma begreppsapparat kan i och för sig tyckas ha varit ett problem men har givit oss många positiva effekter. Det har fått oss att få en mer bred förståelse kring vår forskningsfråga. Att mötas över ämnesgränserna är en av de mest spännande uppgifterna inom informatikområdet.

3.6.8 Kriterier för vetenskaplighet

Allmängiltigheten i undersökningen styrs i den kvalitativa ansatsen av att forskaren kan motivera att urvalet besitter en heltäckande och djup kunskap om problemområdet menar Halvorsen, (1992). Urvalskriterierna för våra intervjupersoner styrdes dels av personlig kännedom om personerna genom en av uppsatsförfattarnas arbete, dels av vår övertygelse om att de har allmängiltig kunskap inom problemområdet.

Giltighet (validitet), är ett mått på hur väl forskaren får rätt bild av verkligheten vilket i sin tur beror på val av lämpligt mätinstrument (Lundahl & Skärvad, 1999). Genom de personliga intervjuerna är det lättare att bilda sig en uppfattning om det är rätt bild som förmedlas av de intervjuade personerna. Detta kan man bl a göra genom att ställa följdfrågor, be om exempel, betrakta hur respondenten agerar under intervjuns gång. Vi har uppfattat våra respondenter som ärliga och mycket öppenhjärtiga.

Inre validitet (innehållsgiltighet), uppnår man om frågorna mäter det man föresatt sig att mäta (Wiedersheim-Paul & Eriksson, 1999). Vårt val av intervjuer ger en hög inre validitet anser vi, med hänsyn till att färre missförstånd kan uppstå då man som intervjuare kan göra förtydliganden av frågorna samtidigt som svaren kan följas upp direkt för att få stämma av om man uppfattat korrekt. Om det finns en överensstämmelse mellan problemområdet och svaren från respondenterna uppnår man inre validitet.

Pålitlighet (reliabilitet) får man genom att forskaren svarar upp till fastställda syften samt att mätningen inte påverkas av vem som utför den eller under de omständigheter som den sker (Wiedershiem-Paul & Eriksson, 1999). Vi anser att pålitlighetens värde ökar om man utförligt beskriver tillvägagångssättet för insamling, bearbetning och analys av det insamlade materialet.

Noggrannhet innebär att forskaren är samvetsgrann för att komma rätt i frågeställningarna under intervjuerna och att förhålla sig objektiv är viktigt och därför är det vid intervjutillfället viktigt att ställa icke-ledande och opartiska samt förutsättningslösa frågor (Wiedershiem-Paul & Eriksson, 1999). Att vara fullständigt objektiv är inte möjligt, men däremot är det fullt möjligt att vara medvetet subjektiv och ärlig. Det man bör eftersträva, vilket vi också har gjort och haft i åtanke vid under insamlingen av våra empiriska data samt i analysarbetet, är att istället eftersträva största möjliga saklighet.

3.6.9 Alternativ metod

Ett alternativt sätt att samla in material kunde varit att enbart genomföra intervjuer. Som vi beskriver i avsnitt 2.5 om kravprocessen är intervjuer ofta det primära sättet för att samla in information om användarnas krav även om det är bra att också använda sig av prototyping för att få fram sådana krav som inte intervjuerna förmår att täcka in. Vi bedömer att det kunde ha gått relativt bra att nå en grundläggande förståelse med enbart hjälp av intervjuer, men en lika täckande bild och förståelse hade varit mycket svårt att få enbart med hjälp av intervjuer. Processen hade säkerligen dragit ut på tiden och fler respondenter behövts om resultatet skulle bli likvärdigt.

Vidare kunde ett tillvägagångssätt vara att först göra intervjuer för att få ett underlag till att arbeta fram en prototyp. Prototypen skulle därefter kunna itereras och diskuteras med domänexperterna. Vi tror dock att det hade vållat stora bekymmer att med hjälp av intervjuer förstå varandra och kommunicera eftersom risken är överhängande att vi pratar om olika saker. Detta förfaringssätt skulle

inte i samma utsträckning ge möjlighet att förklara och diskutera respondenternas svar. Det är mycket enklare att prata utifrån en ”färdig” prototyp. Dessutom skulle respondenterna haft svårare att förstå vår idé och därför skulle de säkerligen ha svårare att lika träffsäkert svara på intervjufrågorna. Detta för våra tankar till ett klassiskt problem som belyses av en respondent, *”att man glömmer tillräckligt lära känna användarens krav och behov”*, användarna representeras i vårt fall av domänexperterna.

4 Resultat

Uppsatsens resultat redovisas dels under avsnitt 4.1 "Prototyping", där utvecklingen av den mellanliggande programvaran presenteras efter hand som iterationerna framskridit. Dels redovisas under avsnitt 4.3 "Intervjuer", resultaten av de intervjuer som genomförts med domänexperterna. Resultatdelen är ett relativt rent återgivande av vad arbetet hittills gett i prototyp- respektive intervjudelen av arbetet. Att knyta ihop detta till en helhet gör vi i avsnitt 5, diskussion och slutsatser.

4.1 Iterationer med tidiga utvecklingsgruppen

Diskussionerna i den tidiga utvecklingsgruppen kan delas in i några klart urskiljbara frågor:

- Vilken funktionalitet den mellanliggande programvaran skall ha.
- Vilka moment som ingår i och i vilken sekvens en ODBC-koppling upprättas.
- Hur frågor om säkerhet och åtkomsträttigheter påverkar utformningen.
- Vilka konfigureringar av systemet som är nödvändiga.
- Grafiskt gränssnitt.

4.1.1 Första prototypen

Enligt INUSE [2] är det möjligt att arbeta med flera slag av prototyper och typiskt gör man den första själv. I den första prototypen vävdes de tankar och idéer in som vi själva fått under det inledande arbetets gång genom litteraturstudier, referensapplikationer och inte minst egen reflektion.

Den första prototypen utvecklades i PowerPoint, verktyget ger möjlighet att snabbt ta fram ett enkelt förslag till hur funktionalitet och gränssnitt kan byggas upp. I det tidiga skedet var en av prototypens viktigaste funktioner att förmedla idén bakom RFID-Connect, i första hand gällande funktionalitet, i andra hand gällande gränssnitt. Prototypen är uppbyggd med en funktion per fönster, detta för att snabbt kunna lägga till och dra ifrån delar.

Figur 16. Startsidan till RFID-Connect version 1.

4.1.2 Första iteration

Prototyp 1 itererades med två före detta studenter med goda kunskaper inom för det här arbetet relevanta områdena: RFID, protokoll för RFID och programmering. Diskussionen kring prototyp 1 gav följande förslag till förändringar/konstateranden/frågor att fortsätta med:

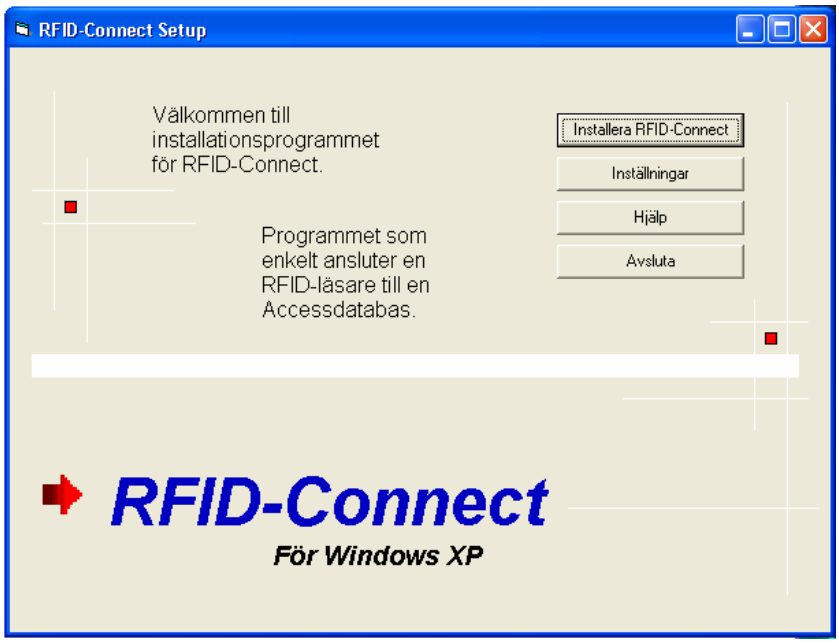
- En ODBC-koppling måste göras för att önskad funktionalitet skall uppnås.
- ODBC-kopplingen skall namnges.
- I den databas man kopplar mot måste en kolumn väljas där matchande RFID-identiteter finns. Det är önskvärt att möjliga kolumner listas upp automatiskt under konfigureringen.
- En diskussion kring var funktionalitet primärt skall läggas, i RFID-Connect eller i databasen?
- Ett preliminärt ställningstagande är att så mycket funktionalitet som möjligt skall läggas i RFID-Connect, utom där motsvarande automatiska funktion finns i databasprogrammet. Alternativet är att användare skriver kod i databasprogrammet men det bedömdes som önskvärt att vanliga funktioner skall kunna hanteras av RFID-Connect.
- Programvaran bör göras tvådelad, en installationsdel där merparten av systemets konfigureringar görs, och en servicedel som är den del av programmet som utför de tjänster som efterfrågas.
- Följande val skall göras i samband med konfigurering/installation men går naturligtvis att ändra:
    - Hur RFID-Connect skall startas under vanlig drift, manuellt eller automatiskt.
    - Vilket funktion som skall aktiveras vid känd respektive okänd RFID-identitet.
    - Säkerhetsinställningar, åtkomsträttigheter.
- Nästa prototyp bör skapas i en miljö som möjliggör viss funktionalitet.

Figur 17. Gränssnitt version 1, en av de funktioner prototypen visade - vilka seriella anslutningar systemet har identifierat.

4.1.3 Andra iteration

Samma studenter som deltog första gången deltog var med i denna iteration. Med denna version kom den första prototypen som med mer rätta kunde kallas prototyp. Här fanns viss funktionalitet som innebar att man kunde navigera i programmet och se hur de olika gränssnitten hängde samman och vi fick också möjlighet att titta på kommunikationen mellan program och användare. Från och med denna version av programvaran är prototyperna gjorda i Visual Basic.

Att navigation i programmet nu var möjligt innebar att vi kunde företa de första begränsade användartestningarna. Installationsdelen är såpass realistisk att kopplingen RFID och databas faktiskt skulle kunnat ske.



Figur 18. Startsidan för RFID-Connect version 2.

Diskussionen kring denna version av prototypen gav följande förslag till förändringar/konstateranden/frågor att fortsätta med:

- Ordningsföljden i de olika stegen i ODBC-kopplingen relativt andra moment är mer komplicerad än förutsatt. ODBC-kopplingen måste göras först, vilket är en stor nackdel då detta tycks vara det svåraste momentet för användaren.
- Viss tvekan kring möjligheten att automatisera ODBC-kopplingen råder. Detta bedömdes nu som den absolut viktigaste frågan att reda ut.
- Två alternativ under serviceinställningar bedömdes så lika att det ena plockades bort.
- Navigeringen fungerar inte fullt ut. Viss förvirring kring knappars namngivning och placering. Alternativen: Fortsätt, Tillbaka, Avbryt och Hjälp syntes vara mest relevanta.
- Vilken information användare behöver ha som bekräftelse på gjorda val.
- Alternativen på startsidan behövde förtydligas.

En notering är att vi förutsatt för stora kunskaper kring funktioner i databasprogram hos dem vi itererat prototypen med. Problemet är att man behöver veta vilka tjänster databasprogrammet ger tillgång till vid en koppling som den vi eftersträvar. Den övergripande idén med den mellanliggande programvaran är något svår att kommunicera. Vi behöver ett grafiskt presentationsmaterial som komplement till prototypen.

Figur 19. Prototypen är nu mer programlik. Navigeringen innehåller fortfarande brister.

Oklarheterna kring om ODBC-kopplingen kunde skötas automatiskt eller ej tvingade fram en omfattande hjälpfunktion för att klara momentet. Hjälpen designades som en guide. Guiden ligger framme under hela genomförandet och tillåter användaren att när som helst återgå till RFID-Connect. ODBC-kopplingen är en av de mest fundamentala i den mellanliggande programvaran och utan denna kan inte programmet fungera. Särskilt olyckligt är att den måste ske som ett av de allra första stegen i installationen. Svårigheterna i det här momentet kan inte underskattas, funktionen behöver sannolikt förses med ytterligare hjälp om den inte kan automatiseras. Sessionen gav ingen möjlighet att testa guiden då respondenterna är ytterst förtrogna med ODBC-kopplingar. Svårigheterna ligger inte enbart på nivån att få användaren att göra rätt. Han/hon måste också förstå det mycket abstrakta i varför man gör en sådan koppling.

Figur 20. Guiden för ODBC-koppling.

4.1.4 Tredje iteration

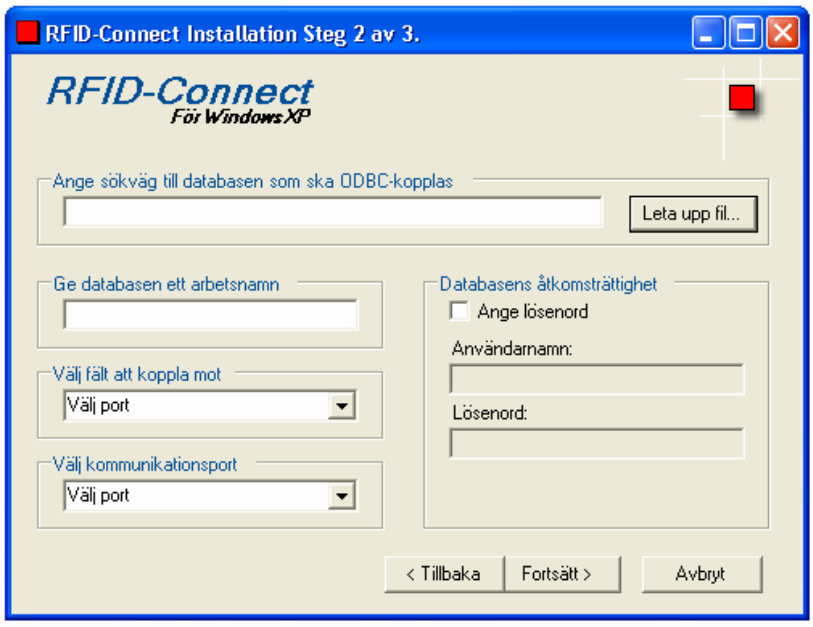
Denna version itererades med en för arbetet ny respondent. Denne har en annan bakgrund än tidigare respondenters då hans kunskaper ligger primärt inom området databaser och även till viss del inom programmering. Det viktigaste resultatet, och egentligen enda skälet till denna session var att få klarhet kring ODBC-kopplingen. Det helt entydiga beskedet var att denna koppling utan större svårigheter kan automatiseras.
Diskussionen kring denna prototyp gav följande förslag till förändringar/konstateranden/frågor att fortsätta med:

- ODBC-koppling kan automatiseras.
- Vi måste hantera åtkomsträttigheter på ett adekvat sätt. Ligger måldatabasen i ett nätverk är den och värddatorn sannolikt lösenordsskyddade.

Figur 21. Automatiserad ODBC-koppling innebar lättare handhavande.

Efter denna iteration bedömdes prototyperna såpass mogna att den inledande itereringen kunde avbrytas. En iteration med starkt fokus på gränssnitt och navigering bör komma först efter det att funktionaliteten specificerats närmre.

Ytterligare en version gjordes på samma iteration, i ett alternativt gränssnitt bland annat påkallat av tillgång till en ny utvecklingsmiljö.



Figur 22. Ett icke-itererat gränssnittsförslag av version 2 Framtaget med .NET.

4. 2 Iterationer med domänexperter

Med domänexpertgruppen fick frågorna och arbetet en delvis annan inriktning, här handlade diskussionen kring prototypen mycket om:

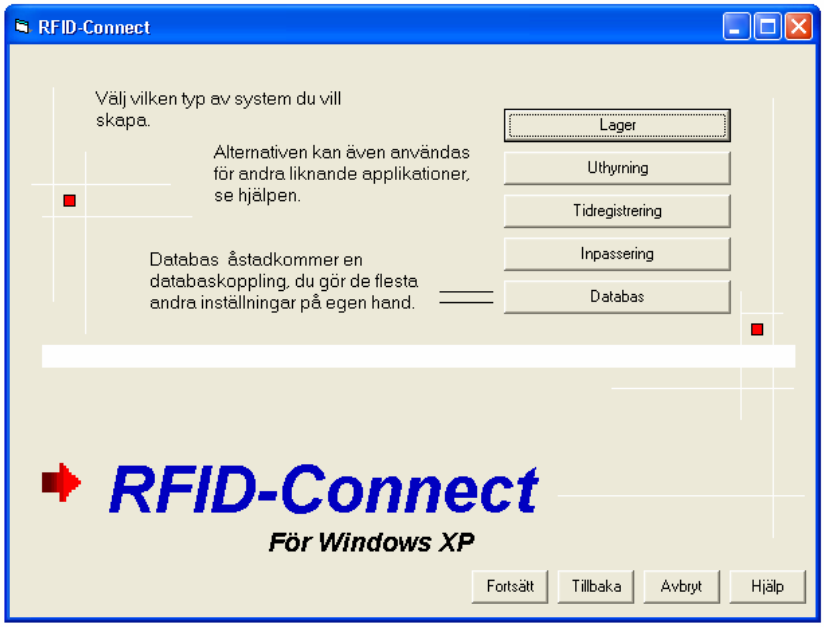
- Vilka databasprogram som går att köra mot.
- Hur tänkta användare kan komma att använda programvaran.
- Hur generell en mellanliggande programvara kan göras.

Följande versioner av prototypen är de itereringar som gjorts med domänexperterna. Tidigare itereringar har syftat till att få fram en mogen prototyp inför intervjuerna med expertgruppen. I tidigare itereringar har prototypen stått i centrum, från och med nu står en diskussion kring prototyp, funktionalitet et cetera i centrum. Bland annat innebär detta att dokumentationen kring itereringarna nu ser annorlunda ut. Tidigare har protokoll förts vid prototypgenomgången, nu spelas samtalen in och skrivs ut. Merparten av de resultat som framkommit redovisas därför under ”Intervjuer”. Några mer konkreta förslag till förändringar har framkommit under intervjuerna, dessa redovisas nedan. De ur prototyphänseende viktigaste resultaten av intervjuerna med domänexperter var:

- Beskedet att den övergripande idén och utformningen av den mellanliggande programvaran har bärighet.
- En diskussion kring hur generell en programvara som RFID-Connect egentligen kan bli. Diskussionen kan enkelt sammanfattas med att det är möjligt att skapa en fullständigt generell mellanliggande programvara. Den tidigare inriktningen om att lägga specifik funktionalitet för databasen i programvaran övergavs och den nya inriktningen på projektet blev nu att göra ett skal som fullt ut bygger på att använda databasens funktionalitet. Det här innebär att funktionaliteten i mellanliggande programvaran i huvudsak blir att installera och konfigurera systemet. Vidare att lyssna på de valda seriella portarna efter inkommande RFID-identiteter och att koppla dessa till en händelse.
- Att flertalet RFID-tillämpningar bygger på att man har fler läsare än vi förutsett.

4.2.1 Första iteration

Första respondenten framförde många tankar kring hur generell en mellanliggande programvara kan vara. Han pekade i huvudsak på två svårigheter i sammanhanget. Den första gäller funktionaliteten, finns funktioner på nivåer i programmet att de påverkar olika slag av tillämpningar var han mycket tveksam. Den andra svårigheten han pekade på var att det helt enkelt kan vara svårt att sälja en för generell produkt. Ett alternativt upplägg på RFID-Connect togs fram, denna gång med tanke att riktas mot olika kundgrupper.



Figur 23. En version av RFID-Connect tänkt för olika typer av tillämpningar.

4.2.2 Andra iterationen

Till skillnad från respondent nummer ett var denne fullt övertygad om att det bör vara möjligt att göra en generell applikation och skulle det vara förenat med svårigheter att göra den så borde man i första hand satsa på en mellanliggande programvara med inriktning på uthyrningsverksamhet.

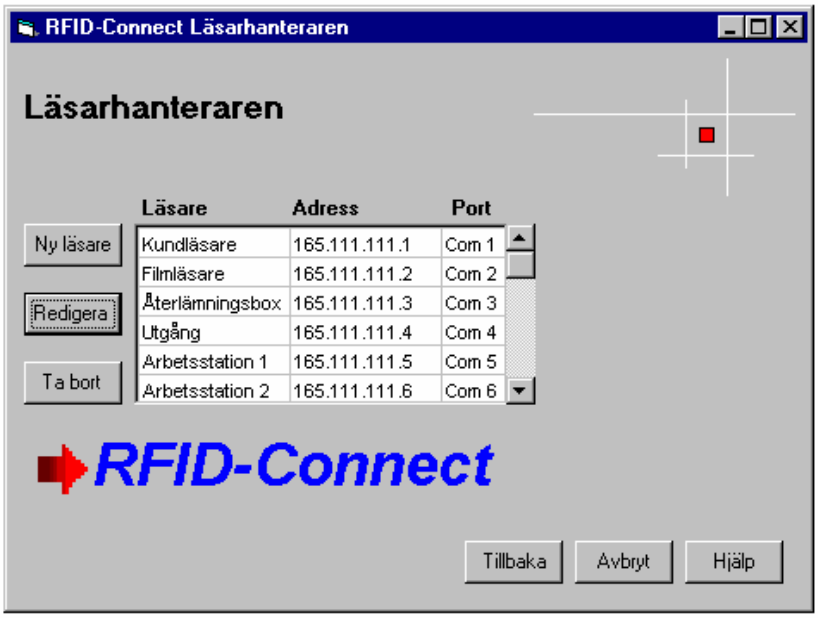
4.2.3 Tredje iterationen

Med respondent nummer tre fick vi möjlighet att knyta ihop mycket frågor och diskussioner som väckts i tidigare intervjuer. Den viktigaste av dessa var att vi kunde konstatera att det är fullt möjligt och förmodligen bättre att utforma RFID-Connect som ett skal som fullt ut förlitar sig på funktionaliteten i databasprogrammet. Den andra viktiga frågan ur prototyphänseende var att vi blev medvetna om att de flesta användare kommer att vilja använda fler läsare än vi förutsett.

Som resultat av intervjun tar vi fram en ny prototyp som bygger på händelsestyrning som en tydlig princip. Centralt i programvaran är en händelsehanterare som dirigerar alla händelser från läsare vidare. I händelsehanteraren namnger användaren en händelse och talar om vad som skall ske när villkoren uppfylls. Händelsen ”Ny kund” i figur 24 beskriver att formuläret Ny kund i databasprogrammet startar, när en okänd tagg kommer på läsaren kundkort, vilket indikerar ett kundkort utan kunduppgifter.

Konfigurering av programvaran sker i en miljö med ett konsekvent gränssnitt med en Läsarhanterare där läsarna i systemet namnges. I ODBC-hanteraren namnger man sin ODBC-koppling, talar om mot vilken databas den skall kopplas och gör nödvändiga inställningar för åtkomst av databas i nätverk.

Figur 24. Händelsehanteraren innebar ett helt nytt händelsebaserat upplägg av RFID-Connect.



Figur 25. I läsarhanteraren namnges läsarna med valfritt namn.

4.3 Intervjuer

Intervjuerna har utförts på plats hos respektive företag för att respondenterna ska känna sig hemma i sin vardagliga miljö. För att resultatet ska speglas utifrån den kunskapsbas de olika respondenterna har med sig sedan tidigare inleder vi med att presentera deras bakgrund som de själva återger den under intervjuerna. Alla svar som inte är direkta citat har skrivits om och återgivits kring de begrepp som vi behandlat i samband med intervjuerna. Resultatet av intervjuerna är med andra ord en blandning av ordagrann återgivning samt en av oss opåverkad spegling av vad alla respondenternas bidragit med utifrån deras tidigare erfarenheter och domänkunskap.

4.3.1 Respondenternas bakgrund och erfarenheter

Den första respondenten har 15 års erfarenhet av systemutveckling och projektledning. Arbetsuppgifterna har huvudsakligen varit lite större realtidssystem i en stor koncern för bland annat övervakning av kärnkraftsverksamhet. Därefter har respondenten även arbetet vid en högskola och där främst sysslat med att bygga upp forskningsverksamhet kring realtidssystem. År 2000 blev respondenten egenföretagare i ett företag som har haft stor framgång och sedan dess har han varit verkställande direktör för företaget på heltid. Utbildningsmässigt har respondenten civilingenjörsexamen i maskin och datorteknikområdet samt en teknisk licentiat i datorsystemteknik med inriktning mot programmeringsutvecklingsredskap i samband med kritiska realtidssystem. Utvecklingen har därefter främst berört och glidit över mot trådlös kommunikation, vilket också innefattar RFID-tekniken.

Den andra respondenten har som tekniker arbetat på ett stort elektronikföretag med främst radarstationer i 7 år. Respondenten har bland annat varit ansvarig för alla licensaffärer utomlands vid nya fabriksetableringar. Därefter började respondenten med konsultverksamhet under några år för att där utföra specifika uppdrag åt kunder baserat på RFID-tekniken, bland annat inom bilindustrin och industrielektronik. De sista 4-5 åren har respondenten främst specialiserat sig inom tidtagningsområdet i samband med olika sporter för att där kunna bygga upp en stabil verksamhet. Vidare är respondenten involverad i internationella samarbete med bland annat företag i Holland där och användandet av RFID-tekniken är i framkant, kanske världsledande enligt respondenten. Utbildningsmässigt är respondenten elektronikingenjör.

Den tredje respondenten började arbeta med RFID redan 1984, i ett företag som var störst i världen på att utveckla styr- och navigeringssystem åt förarlösa truckar. Företaget som respondenten arbetade för ville så småningom renodla verksamheten och den strukturerades om, och därmed kom arbetssysslorna i början av 90-talet mer att handla om informationshantering. Detta innebar att det blev allt mer systemtänkande och mjukvaruutveckling. Så småningom visade det sig att folk inte i första hand ville ha teknik utan funktion. Utbildningsmässigt är respondent 3 tekniker som började som elektronikkonstruktör och är första generationens elektronikingenjör. Fokus blev därefter marknad och han blev platschef och försäljningschef för ett stort elektronikföretag. Utbildningen breddades successivt med bl a 60 poäng ekonomi och marknadsföring samt juridisk utbildning. Respondenten uppskattar utbildningen till ungefär 10 års universitetsstudier inom en massa olika områden med mer bredd än djup.

4.3.2 Intervjuundersökningens validitet och reliabilitet

En förutsättning för att få ett bra resultat och förståelse i denna typ av intervjuer är att vara noga med urvalet av respondenter.

I och med att vi använt oss av prototyping i tidigare skeden av vårt arbete har vi snabbt fått fram en beskrivande prototyp. Denna prototyp har fungerat bra

som ett diskussionsunderlag, och hjälpt oss att få respondenterna att både snabbt och träffsäkert förstå vår idé med en mellanliggande programvara. Vi har i och med prototyputvecklingen även utvecklat vår egen förståelse på ett helt annat sätt än vad som annars hade varit möjligt. Detta har underlättat mycket både beträffande hur effektivt vi kunnat utnyttja intervjutiden och hur effektivt vi kunnat få respondenterna att svara.

Vidare har vi lagt ner mycket möda på att inte färga intervjupersoner med ledande frågor och gester. Vi anser att vi lyckats relativt bra, mycket beroende på vår stora tidigare erfarenheter av att genomföra intervjuer i samband med 4 tidigare genomförda arbeten på C och D-nivå samt långvarig yrkeserfarenhet med samtal som verktyg. En annan aspekt som många underskattar är tiden för intervjuerna som oftast är för snålt tilltagen. Vi har varit noggranna med att avsätta ordentligt med tid så att vi ej behövt stressa under intervjuerna. Detta har resulterat i en hel del intressanta avstickare ifrån intervjuguiden som tillfört oerhört nyttiga och intressanta synpunkter, framför allt har det underlättat för oss att förstå helheten på ett mycket rikare sätt än vad som annars skulle varit fallet.

Det resonemang som vi fört ovan påverkar resultatet i stor utsträckning. Det är viktigt att belysa dessa aspekter för resultatet ska kunna tolkas på ett korrekt sätt och med rätt dignitet. Resultatet kommer vi att återge så opåverkat som möjligt, därav lyfts även en del betydelsefulla citat fram.

4.3.3 Övergripande synpunkter på mellanliggande programvaran

I samband med intervjuerna förhörde vi oss om begreppen som kännetecknar den externa kvaliteten. Nämligen functionality, reliability, usability, effeciency, maintainability och portability. Respondenterna fick dels försöka belysa hur viktiga de respektive begreppen var för att därefter motivera varför, se intervjuguide i bilaga 3. Det har dock varit svårt att få varje begrepp helt klarlagt på det sätt vi tänkt från början eftersom det var en del begrepp som respondenterna inte helt kände till och var bekanta med sedan tidigare.

En sak som framgick under intervjuerna var att alla respondenterna verkade ha väldigt mycket projekt på gång, det händer otroligt mycket på marknaden och efterfrågan verkar vara mycket stor, så stor att de inte hinner med mer än en liten andel av de projekt och idéer som är i omlopp. Detta speglas klart av alla respondenterna på olika sätt och uttrycktes bland annat av en respondent som påpekade att *"drivet efter RFID-tekniken är stor med tanke på att det inte är ett år sedan RFID-tekniken för första gången nämndes i radion, och då med negativ klang"*. Två av respondenterna uttrycker att publicitet kring RFID-teknik främjar marknaden eftersom kunder ofta inte vet om att den finns och går att använda.

Det har inte heller varit lätt att dela in respondenternas svar och synpunkter under de 6 begrepp som ingår i QIU. Vi anser oss ändå ha fått en relativt bra och täckande uppfattning om vad som anses och är viktigast i samband med denna typen av mellanliggande programvara som RFID-Connect utgör. En del

kommentarer och synpunkter berör fler än det begrepp som de är placerade i och kan därför återkomma under mer än ett begrepp, men i så fall bidra i en annan mening.

Vidare har vi även fått utöka med ytterligare rubriker för att kategorisera det som var svårplacerat i begreppen för extern kvalitet i QIU.

4.3.4 Functionality

Functionality är enligt respondenterna något av det mest viktiga att lyfta fram i samband med utveckling av småskaliga informationssystem där RFID används som teknik. Tillsammans med usability framställer respondenterna funktionalitet som en självklarhet, eller som en respondent säger att *"det grundläggande är functionality och usability. Det andra faller ju om inte det fungerar"*. Vidare uttrycker en respondent samma sak med den spontana motfrågan *"går det att bygga en applikation utan funktionalitet?"* medan en annan kommenterar funktionalitet med samma självklarhet, *"det måste ju uppfylla syftet. Köper du en gräsklippare vill du klippa gräs...det är funktion som användare förväntar sig"*.

En annan synpunkt som lyfts fram är att det är mycket viktigt att programvaran uppfyller den funktionalitet som den förväntas ha. Funktionalitet bedöms ha mycket stor relevans i samband med denna typ av mellanliggande programvara. En av respondenterna uttrycker detta på följande sätt:

> *"Användaren förväntar sig funktionalitet. Det är användaren som bedömer om något är bra eller dåligt, även om det teoretiskt, tekniskt uppfyller alla önskvärda krav...tycker inte användaren att man uppfyller kraven blir det inte bra ändå"*.

Kunder kan ej heller uttrycka allt beträffande krav och behov som en respondent utifrån sin professionella roll konkretiserar:

> *"de kan vara jättebra på att plocka plockbitar men känner inte till spelreglerna, det är därför vår uppgift att kunna något om och beakta även sånt som han inte tänkt på. Ett exempel är den lagförändringen som kommer den 1 januari 2005 och som tvingar till vissa saker som att exempelvis kunna plocka fram en uppgift inom 48 timmar"*.

Vidare förs även ett resonemang om hur viktigt det är att utgå ifrån användarens krav och behov, det kommenteras med *"man måste...så långt det går lära känna användarnas förväntningar och behov, det är det här klassiska misstaget man gör..."*.

4.3.5 Reliability

En respondent beskriver att vikten av reliability ligger någonstans mellan liten till stor relevans för denna typ av applikation. Vidare påtalas att *"vi vänjer oss vid vissa saker som t ex när vi använder en telefon och vi ej kommer fram, då försöker vi bara lite senare och på så vis får reliability en något mer nedtonad roll"*. En respon-

dent påpekade även, vid tal om användning av RFID i samband med sortering av tvätt, att:

> *"olika system har olika krav, t ex går det ej att jämföra ett styrsystem i ett flygplan med ett tvätteri där du tvättar kläder. Ett flygplan kan komma att störta om systemet inte svarar inom 15 millisekunder medan det i tvätteriet har föga betydelse"*.

Ett annat exempel som lyfts fram, i samband med reliabilitet, är:

> *"...i ett lager med datorer kan det ha större betydelse att allt fungerar som det ska. Det finns alltid mer kritiska situationer eller mer kritiska system och då kan det vara ett problem. Men om man även beaktar detta är ändå inte tillförlitligheten den allra mest kritiska parametern"*.

En synpunkt som framfördes i samband med att unga killar som hade relativt stort kunnande fick ett RFID-system att krascha p g a att de gick in och modifierade bara för skojs skull, *"ju mer kunnig en användare som hanterar en applikation är, desto större är risken för att han ändrar något som gör att systemet kraschar"*. *"Det fungerar i regel bättre..."*, menar en respondent, *"om användaren är lite mindre kunnig och rädd för att ändra saker som kan få systemet instabilt"*.

Med andra ord är inte reliability alltid så viktigt i samband med denna typ av småskaligt system som man kanske inledningsvis skulle kunna tro.

4.3.6 Portability

En respondent påtalar att portabilitet saknar relevans, och påtalar detta genom att ställa motfrågan *"...ni väl har en CD-skiva för PC?"*.

En respondent föreslår .NET som utvecklingsplattform eftersom det därigenom är lätt att få det att fungera på handdatorer som kör operativsystemet Windows CE. Vidare påpekar respondenten att *"...det är bra att kunna slippa släpa på tunga containrar till en fast läsare, därav vikten av att ha portabla läsare"*.

En annan sak som framkom var att det kan finnas en blandning av offline och online system eller som det uttrycks *"du kan tänka dig att du inventerar telefonstolpar, då kommer du utanför alla täckningsområden...inom täckningsområdet kan du jobba online..."*. Portabiliteten anses i detta avseende vara ganska viktig.

4.3.7 Usability

Användarvänlighet kan sägas vara en av de mest viktiga aspekterna att beakta tillsammans med funktionalitet. Alla respondenterna viktade detta kriterium mycket högt. En av respondent menade att *"allt annat är en avhängigt av usability, fungerar den biten kommer det andra mycket av sig självt"*.

Det påtalas även att ett system av detta slag måste vara användarvänligt så det går att applicera det över huvudtaget, t ex genom att lägga in fält i databasen

som man vill eller t ex genom att konfigurera rapporter för att få ut data på rätt sätt, och påpekar att *"får man detta att fungera har man kommit långt"*.

Bland annat uttrycks även att användarna vill ha enkla system, men att det också kan finnas vissa nackdelar med alltför enkla system:

> *"de som är ovana vid användningen av sådana här system vill bara kunna mata in saker för en sak och tycker systemet är jobbigt att jobba med. Skall man ha ett väl fungerande system ska man också förse systemet med alla parametrar som systemet behöver"*.

Användarvänliga system anses som mycket viktigt och det bör som respondenten uttrycker det vara *"näst intill självinstruerande"*. Detta nämndes även av en annan respondent som tryckte på att applikationer bör göras så enkla som möjligt.

4.3.8 Efficiency

Här svarar respondenterna relativt kort. En menar att efficiency-kraven måste uppfyllas *"annars så köper kunden inte systemet"*, medan en annan respondent förklarar att *"detta är inte ett prestandasystem"*. Respondenten förklarar vidare att:

> *"...tänk er att vi jämför med Wallmarts läsare i sina lager, där det sitter en tagg på varenda conflakespaket. Det är en helt annan division jämfört med kåren som skulle räkna sina kanelbullar"*.

Respondenten menar att det är stor skillnad mellan storskaliga och småskaliga användare - effektiviteten har därvid inte samma framskjutna roll som i denna typ av småskaliga system.

4.3.9 Maintainability

En respondent uttrycker att det är väldigt viktigt att tänka på att programvaran är lätt att underhålla för annars *"är det lätt att den förfaller och ramlar i graven"*. Vidare leder resonemanget in på att relevanta data, som kan bli mycket omfattande, även ska kunna nyttjas för något ändamål. En respondent tyckte att *"om man loggar något med taggar kan man få hur mycket data som helst egentligen...jag tror säkert att någon som går igång med ett litet projekt spinner loss, för då får de så mycket data - att oj..."*, en annan uttrycker det som *"...en annan fråga vad som ska hända med all data som samlas in eftersom den kan bli väldigt omfattande"*. Vidare framgår det att för att systemet ska fungera gäller det att hela tiden underhålla det.

En annan respondent ger maintainability stor betydelse av samma anledning som usability, *"att den måste finnas där"*. Samtidigt påpekar respondenten att det kanske inte stämmer i vissa avseenden när det gäller tekniken, *"ofta byts en generation boxar ut mot nästa generation. Det är klart att det måste vara kompatibla och*

fungera...". Vidare påpekas det även att det kan finnas en supportorganisation som hanterar maintainability.

Det påtalas att man naturligtvis måste kunna underhålla och framför allt vidareutveckla programvaran eftersom förutsättningarna förändras för användaren under tidens lopp. Det uttrycks bland annat särskilt tydligt av en respondent som att *"förutsättningarna förändras för användaren under tidens lopp...man växer i insikt och kunskap och kommer själv på andra användningsområden och tillägg och förändrad funktionalitet"*.

4.3.10 Programvarans förutsättningar att uppfylla sitt mål

Två respondenter uttrycker rakt ut att implementeringen kan underlättas av en mellanliggande programvara, bland annat nämner en av dem att *"implementering skulle kunna underlättas av ett sådant här system eftersom användarna idag som regel behöver hjälp att få det att fungera"*.

Det är svårt att veta vad kunden förväntar sig av ett system. Detta är något som uttrycks speciellt tydligt hos en respondent som påtalar att *"det går inte att föreställa sig vilka knepiga applikationer och specialfall som olika kunder kan ha"*.

4.3.11 Generaliserbarhet

Alla respondenterna var väldigt bestämda att varje kund som de har alltid har olika behov att tillgodose beträffande hur ett system ska fungera och se ut. Detta beskrevs på alla möjliga olika sätt och återkom ständigt under intervjuerna eller som det lyftes fram - hur generell en sådan här mellanliggande programvara kan vara beror på vad kunden använder den till.

En respondent säger att *"om man ska göra en applikation generell så får man ha med vissa kriterier och säga att det här fungerar, vilket leder vidare till anpassning i olika nischverksamheter"*. Nischverksamheter påtalas av alla tre respondenter som en möjlig lösning, vilket en respondent beskriver på följande sätt:

> *"en utveckling som sker successivt allt eftersom en produkt på marknaden mognar...i början kan det handla om en relativt smal produkt som successivt kan byggas ut med andra nischer allt eftersom nya projekt och anpassningar utvecklas...det är tufft att bära alla kostnaderna för att utveckla en applikation som tillhandahåller all funktionalitet för flera olika nischverksamheter från början"*.

En tanke som alla respondenter alltså talade för var att det skulle vara möjligt att nischanpassa sådana här system, t ex *"ett grundsystem som anpassas för exempelvis butiksändamål"*. Respondenten exemplifierar genom att tala om *"streckkodsläsare som läser in produkter, där samma sak händer varje gång"*.

En respondent påpekar att *"det kan vara väldigt svårt att göra en programvara som fungerar både för en rörfirma och en uthyrningsverksamhet, det är alltid så att man får snickra lite själv"*. En respondenten nämner att *"kunden vill alltid ha något annat än*

vad som finns och det går aldrig att göra något generellt". Men vidare påtalas att *"det nog skulle fungera om kunden får välja layout och logga samt text där och där"*. Men samtidigt påtalas att *"folk är så besvärliga, särskilt kunder"* och det framgår av en respondent att det är sällan som de gör något generellt utan istället *"hoppar de rakt på"* eftersom *"det är så mycket lättare att göra så"*.

En annan respondent uttrycker sin förvåning över graden av återanvändbarheten i denna typ av underliggande systemlösningar; *"Vi har blivit överraskade av återanvändbarheten i de generella plattformar vi har byggt när vi har specialiserat oss mot olika nischer"*.

Till skillnad ifrån en helt generell mellanliggande programvara nämns även en annan aspekt i samband med en generell branschspecifik programvara. Användaren skulle kunna göra kosmetiska förändringar i ett skal ovanpå den för branschen generella programvaran.

Det påpekas även att:

> *"en sak som försvårar generaliseringen är att varje kund har sin egen nomenklatur även fast de kan befinna sig i samma bransch. Detta innebär att det talar för en lösning där användaren kan anpassa sitt gränssnitt till stora dela precis som denne vill ha det. Kunden kan då konfigurera lite som han vill på toppen men inte ändra i huvudstrukturen, men han får utseendet som han vill ha det"*.

För att kunna anpassa mot kund föreslås alltså ett skaltänkande med olika skikt vid utformning av systemet vilket uttrycks på följande sätt *"det underliggande systemet kan säkert göras relativt generellt, men det behövs ett skal för att anpassa systemet mot användaren på toppen"*.

En annan strategi som lyfts fram kan vara att anpassa systemet efter någon uppgift eller mål, t ex att hantera lager vilket säkert inte bara små användare med relativt begränsat antal produkter skulle kunna nappa på.

4.3.12 Tänkbara möjligheter och användningsområden

En respondent påpekar att man måste tänka även på marknaden, *"ur marknadsföringshänseende är det bra med en nisch eftersom det är mycket lättare att marknadsföra inom en sådan. Den lille småhandlaren är ingen homogen marknadsnisch"*.

Det kan vara svårt att få kostnaderna att motiveras mot kund, taggarna kostar och läsarna likaså. Ska man sedan bygga ett Radio-LAN är det också något som måste tas med i kostnadsberäkningen. En respondent uttrycker detta på följande vis:

> *"vi försöker titta på allt beträffande RFID-tekniken, men sen har vi också målgruppen eller kunden att ta hänsyn till..i företagarens värld kan man*

> *hitta häftiga lösningar med handdatorer hit och dit, men vem är målgruppen och vem betalar för allt?"*.

Ett problem som man alltid råkar ut för är läsavståndet, *"folk tror att de kan läsa av saker hur långt bort som helst. När det väl inser det rensar det även de olika saker man kan använda RFID till också"*. En respondent nämner avläsningsproblematik beträffande trucklaster med ofta stora skrymmande föremål som inte heller är helt enkla att läsa av. En annan respondet belyser problemet genom att klargöra att *"med våra jätteantenner har vi ett läsavstånd på 60 centimeter för de passiva taggarna och om det är större taggar rör det sig kanske om ett par meter"*.

Många kunder har inte en aning om vad RFID är, vilket påtalas av alla respondenter. Men de skulle kunna ha stor nytta av tekniken. Det gäller att titta utifrån kundens perspektiv för att hitta användningsområden som de skulle kunna ha nytta av.

Tänkbara användningsområden som lyfts fram är t ex mindre uthyrningsverksamheter som kläduthyrning, videouthyrning. Även intern utlåning av saker, akthantering och tidspassning är något som omtalas som ett mycket lämpliga användningsområden. Vidare nämns även användningsområden som enklare lagerhantering och loggning av olika saker, t ex saker passerar in och ut genom dörrar. I samband med intern utlåning lyfter två av respondenterna lyfter fram att det är speciellt viktigt när det är prylar som har ett stort värde och/eller informationen om dem är värdefull eftersom det är då kostnaderna för tekniken är lättare att motivera.

Lagerhantering med RFID-teknik nämns som *"inte lätt att lösa med denna typ av teknik...lagerhantering av lite enklare eller begränsad karaktär skulle nog gå att lösa"*. En annan respondent belyser svårigheterna genom att visa på att *"vi har så mycket annat att ta hänsyn till i ett lager som skapar svårigheter, som exempelvis läsavstånd, hur mycket lasten på ett visst ställe väger så inte överbelastning sker och säkerheten äventyras"*.

Hantering av tidtagning och tidsstämpling för bland annat timdebitering är exempel på tillämpningar när ett sådant här system skulle passa väl. Bland annat nämns uthyrning av tennistider. En inte oviktig aspekt vid många slag av uthyrning är status på det man hyr ut, exempelvis ledig och klar för uthyrning, ledig men inte klargjord för uthyrning, reserverad osv. En respondent exemplifierar det hela med att:

> *"det kan vara så att man vill hyra något från den 23:e till den 29:e juni...Här är också vanligt att någon typ av status för sakerna som hanteras brukar användas. Exempelvis så kan något vara utlånat, klart för uthyrning, eller återlämnat men inte klart för uthyrning"*.

Vidare nämns också att *"en möjlighet är att man vill hålla koll på hur många timmar något används med tanke på service, etc"*. Detta resonemang återkommer hos två av respondenterna.

Andra exempel som nämns och skulle kunna fungera är loggning av varor i lager eller affärer där varor som passerar olika dörrar loggas och övervakas och på så vis kan utlösa olika larm om något inte står rätt till. En respondent nämner även att *"man kan även tänka sig parkeringshus, kiosker, reklambranschen"*.

En annan möjlig tillämpning är som en respondent uttrycker det:

> *"om vi begränsar oss till RFID så är det mycket det som vi kallar intern uthyrning som exempelvis typ lager, industriskolor. Intern förrådshantering på exempelvis ett sjukhus med röntgenapparater och annat. Det är mängder med saker som är i omlopp och man vill veta var de är. Vi har även det där med kostnadsställe eller inventariehantering samt hur mycket något är använt och sådana saker"*.

Vidare kan ett användningsområde även vara akthantering, som uttrycks av en respondent nedan:

> *"ett annat område är det här med dokument och akthantering. Det finns mycket mappar som ligger i olika rum och så vill man hålla koll på dem för att slippa lägga massa tid på att leta efter dem. Dokumenthantering är relativt enkelt för där hanteras inte kostnadsställen eller koppling till affärssystem och sådana saker"*.

4.3.13 Tänkbara funktioner

De funktioner som huvudsakligen lyfts fram är identifiering av saker och att leta efter saker. Detta nämns i mer eller mindre utsträckning av alla respondenter. För att kunna leta efter saker måste dock läsaren vara handburen. Det påtalas av att det är viktigt att frigöra sig ifrån bilden av streckkodsläsare som kopplas in på datorn och läsare som på ett eller annat sätt är fast monterade.

Att det finns behov av att leta efter saker är viktigt belyses tydligt av bland annat en respondent som varit involverad i bland annat projekt mot byggindustrin där det låg otroligt mycket pengar i att snabbare och säkrare kunna hitta saker som prefabricerats i samband med leveranser vid vissa tider. Respondenten påtalar att *"ofta finns här avtal och viten om inte leveranser sker vid rätt tidpunkt eftersom ofta dyra maskiner som byggkranar och annat kan vara inhyrda för att få dessa prefabricerade element på plats"*.

En annan respondent förklarade att *"de vanligaste frågorna är om en grej finns i den här högen eller om något ska letas fram på lagret. Då associeras identiteten för något i databasen och sen går man ut med läsaren och letar upp matchande identitet vilket är en typ av funktion"*. Respondenten uttrycker även att *"det är lite annorlunda med logistikapplikationer där man vill kunna spåra eller veta var en produkt befinner*

sig just nu och där passerar de snarare en läsare som på så sätt loggar var någonstans var produkten just nu är".

Vidare påpekar en respondent att *"om vi ska leta efter grejor på detta sätt så måste allt som vi letar efter vara känt. Det innebär att allt måste matas in i t ex en databas för att detta ska kunna fungera. Att vi måste registrera varenda grej"*.

En annan sak skulle som lyfts fram är associationer mellan olika taggar vilket uttrycks av två respondenter och nämns av en av dem som att *"man vill veta om två saker hör ihop"*. Ett exempel som lyfts fram i samband med bilreparationer är att *"kolla om versionen på programvaran som styr en bilmotor exempelvis passar den nya reservdel som ska monteras in"*.
En annan funktion som nämndes skulle kunna vara att räkna saker, "t ex hur många av en sak som finns på en hylla, att automatisera lagersaldon".

Vidare påpekas vikten av att lagra saker i en global databas för användningen i samband med inläsning av ett RFID-nummer. Det påtalas bland annat att *"ett nummer säger ingenting om det inte kopplas till något"*. Detta leder även det in på funktionen att taggar måste kunna registreras för att kunna knyta någon information till dem.

En respondent uttrycker även att

> *"en funktion är att göra data åtkomliga för andra system som t ex ett affärssystem. Att andra system kan gå in och hämta den data som de behöver. Beroende på vad ett affärssystem vill ha kan textfiler eventuellt behöva skapas som är formaterade på enligt det för affärssystemet rätta strukturen"*.

4.3.14 Andra synpunkter

På frågan om det finns plats för ett sådant här system svarar en respondent att det finns det med den reservationen att användaren ska känna sig hemma i det system som ska användas. Flera av respondenterna påtalar att, som en av dem uttrycker, *"det är viktigt att sätta sig med branschfolket"*.

En respondent trycker på att enkla system är viktigt, *"systemet måste vara enkelt från början så att man inte lappar och lagar för mycket under resans gång"*. En aspekt som uttrycks är att *"det är viktigt att lösa det som avses lösas, inte allt annat, vilket ofta är mycket frestande. Grundbulten måste vara att man är klar över ett scope och sen håller man sig till det"*.

En svårighet är att *"kunderna kanske inte har tillräcklig kunskap kring hur databasen ska sättas upp eller hur de fungerar"*. En respondent trycker på att *"det är viktigt att i så fall se till att databasen konfigureras upp på ett bra sätt från början"*. Detta påtalas även av en annan respondent som uttrycker det på så vis att *"användaren skulle kunna göra ändringar av systemet av mer kosmetisk karaktär snarare än att gå in och ändra i databasens grundstruktur"*.

Alla respondent påtalar att det bör vara möjligt att byta ut vilken databas som man vill använda. Som exempel nämns att *"Access bör kunna bytas ut mot SQL-server som dominerar i denna typ av lösningar. Access är i och för sig bra eftersom det är tillgängligt genom Officepaketet som alla har"*.

En sak som nämndes av en respondent var att det kan finnas flera olika läsare i samma system. Han påtalade att *"det är lite väl småskaligt att bara ha en läsare även i dessa småskaliga system. Man kan säga att kombinationen läsaridentitet och taggidentitet genererar en viss typ av händelse"*. Som exempel nämndes en läsare på in- och utsidan av en dörr för att registrera om något kommer in respektive ut.

En liten grej som påtalas i samband med valet av taggar är att de vi har valt är rätt dumma. Det vore enligt en respondent lättare om vi valde en tagg som även kunde lagra text i taggarna. Den texten skulle kunna användas för att associera till olika fält i databasen vilket kanske skulle kunna underlätta.

Många av dessa typer av RFID-system är helt transparenta, de tuffar i bakgrunden och syns egentligen inte för användaren, som det påpekades *"det är inte så att man alltid får upp något på skärmen och skall göra någonting"*.

Ett problem som påtalades är att kostnader för att utveckla programvara är mycket stora. Det framgår, särskilt av en respondent, *"att hårdvaran är enkel att köpa in, men det är lätt att lägga det femdubbla beloppet på att utveckla programvaran som egentligen kan utvecklas hur länge som helst"*.

En fråga som förr eller senare kommer är att man vill koppla till exempel uthyrningsverksamheten till ett affärssystem för att kunna fakturera någonting. Här nämner en respondent att

> *"det finns således en kommersiell del i det hela, för kan man ej ta betalt är det inte lika intressant för kunden...ofta talar man initialt om möjligheten till generering av rapporter, men det brukar alltså mynna ut i att man vill ta betalt eller koppla ett kostnadsställe"*.

Vidare är ofta enligt en respondent *"värdet på något det som man vill hålla koll på. Det kan vara produkter på hyllan, något som hyrs ut och detta är grunden med affärssystemen eftersom de har utvecklats utifrån bokföringssystem i grund och botten"*. Vidare beskrivs samma behov av att koppla affärssystem eller rättare sagt göra data åtkomlig i samband med beräkningar.

En sak som framgår av intervjuerna är att kunden inte alltid är medveten om är de lagar och andra förutsättningar som gäller för olika branscher. Behoven som behöver tillgodoses skiljer sig alltså starkt mellan olika typer av branscher. Ibland behövs spårbarhet som det av en respondent nämns *"av t ex inbyggda motorer"*. Ett annat exempel som nämns är

> *"i ett bageri...så måste man hålla koll på grejor...vad man haft i den här degen...vad det är för ursprung på margarinet och vilken batch det kommer ifrån hos tillverkaren...är det gift i...då är det någon som blir förbannad på oss och den batchen"*.

Ett problem som man kan råka ut för är att man ofta går från manuell metod till automatisk. Det kan innebära att man som det framgår av en respondent *"har sparat en man som man måste skicka iväg"*. Det kan således finnas personalpolitiskt känsliga frågor.

Beträffande problem med integritet påtalade alla respondenter att det är ett problem som måste beaktas. En av dem uttrycker det som *"nästa pilsner som kommer att drabba det här området är integritetsproblematiken. Det är väldigt känsliga frågor det där"*.

En annan möjlig utveckling av RFID-tekniken som sägs vara på gång är att RFID-taggar ska kunna placeras i exempelvis färgen på en mjölkförpackning. En av respondenterna tror att det är i och med detta det stora genombrottet kommer att komma.

En alternativ fungerande lösning till RFID som påtalas är streckkoder. För dessa finns det uppbyggda artikelregister som kan nyttjas. Det är ju relativt billiga jämförelsevis och det som skiljer dem ifrån RFID är att RFID unikt kan identifiera en sak snarare än en artikel. Det måste enligt en respondent finnas en poäng med att unikt kunna identifiera en sak.

5 Diskussion och slutsatser

Vi reflekterar över arbetets upprinnelse och resultat. Det huvudsakliga upplägget av den föreslagna mellanliggande programvaran beskrivs. Vidare diskuterar vi några begrepp som förts fram i det tidigare arbetet, bland annat frågan om hur relevant det är att fortfarande diskutera småskalighet. Kapitlet och arbetet avslutas med att vi pekar på några frågor som vi upplever som intressanta för fortsatt forskning.

5.1 Från reflektion till resultat

Varför används inte en teknik i stor utsträckning när den har alla förutsättningar att åstadkomma bra tjänster? Ungefär så löd den första frågan som blev upprinnelsen till den här uppsatsen. Vi båda författare har i olika sammanhang stött på, arbetat med, reflekterat över och pratat om RFID. Otaliga gånger i de här sammanhangen har vi kunnat göra samma konstaterande, människor tycker det här är både intressant och användbart, men tillämpningarna är ännu så länge ganska få.

Vi har i uppsatsens inledning pekat på några som vi tror viktiga förklaringar till varför tillämpningarna ännu är ganska få. En förklaring gäller orealistiska förväntningar på läsavstånd, en annan kan vara besvikelsen över insikten att det inte går att hålla koll på allt, överallt, vid användning av RFID. Kanske är det helt enkelt är för dyrt. Som vi fört fram tidigare är det informationens värde som styr vilka kostnader man kan ta på sig, kanske är denna svår att sätta ett pris på.

Någonstans bland alla frågor om förväntningar, läsavstånd, technology acceptance, integritetsproblem, konstaterande om avsaknad av standardprogramvaror, brist på standarder mellan taggar, olika frekvenser et cetera började vi fundera kring hur man gör de här, som vi tycker, intressanta informationssystemen. Snabbt insåg vi att det var förenat med mycket arbete, det var kunskapsintensivt och multidisciplinärt. Vilket i och för sig inte skiljer ut det från många andra IT-tillämpningar. Det är ur dessa konstaterande man skall förstå hur vi hamnade i forskningsfrågan för detta arbete. Kan man göra något som underlättar implementeringen av småskaliga RFID-system, måste alla börja från noll, vad kan automatiseras?

Vi tror inte vi har löst de problem som eventuellt finns, däremot har vi pekat på en som vi tror fullt realistisk idé om en programvara som gör implementeringen lättare. Alldeles säkert har vi inte nått vägs ände i arbetet, det kommer säkert att dyka upp nya intressanta vändningar framöver. Vår egna stora upplevelse av uppsatsarbetet är nämligen hur inriktningen hela tiden har svängt, ju mer vi undersökt och lärt.

Från början vara tanken att skapa en mellanliggande programvara som själv skulle hysa mycket av den funktionalitet som den nu föreslagna lösningen hanterar via databasprogrammet. Såhär i efterhand var det en mycket ambitiös

tanke som var näst intill omöjlig att klara av. Vi fick kommentarer från domänexperterna som *"kunden vill alltid ha något annat än vad som finns och det går aldrig att göra något generellt"* . Några av anledningarna till att denna lösning ej är möjlig är alltså att kunderna har så olika verksamheter, så olika behov och så olika nomenklatur.

Efter ett tag växte en branschspecifik applikation fram som en möjlig lösning. Det är mer realistiskt med tanke på att verksamheterna inom en bransch är långt mer lika varandra och de använder huvudsakligen samma nomenklatur. Ett problem här vara att det även finns en del företagsspecifika saker som också måste tillgodoses efter kundens önskemål. Ett annat problem var att det är svårt att bära kostnaderna för att skapa alla de branschspecifika lösningar som skulle behövas i applikationen från början. Snarare får det, i fråga om denna lösning, handla om en utveckling av branscher allt eftersom kunder i olika branscher dyker upp så att kostnaderna kan bäras. En branschspecifik lösning skulle kunna fungera, men av respondenterna att döma var det ändå rätt så svårt eftersom företag inom samma bransch också i viss mån skiljer sig åt. Ett skaltänkande växte fram där de stora möjligheterna ligger i att hantera databasprogrammet. Alla respondenterna lyfte fram skaltänkandet som kommenterades bland annat med att *"det behövs ett skal för att anpassa systemet mot användaren på toppen"* och att *"kunden får välja layout, logga samt text där och där"*.

Den branschspecifika lösningen vållade oss bekymmer. Vilka branscher var mer intressanta än andra? Frågan belystes av svaren under intervjuerna genom motfrågor av typen *"vem är målgruppen"*? Kanske skulle vi, som det framkom av intervjuerna, tänka målinriktat snarare än branschspecifikt? Resultatet kändes som ett steg tillbaka och rätt så långt ifrån drömmen om en generell programvara som främjar implementeringen. Vad kunde då göras generellt och vad var gemensamt för alla branscher?

Någonstans under intervjuerna föddes slutligen idén om händelser och en händelsehanterare. Kanske föddes denna tanke när en respondent lyfte fram att *"det är lite väl småskaligt att bara ha en läsare i dessa småskaliga system. Man kan säga att kombinationen läsaridentitet och taggidentitet genererar en viss typ av händelse"*. Händelser är något som varje RFID-system genererar. Något läses in och något ska hända. Vidare hade vi under en tidigare intervju (Svensson, 2004) diskuterat en mellanliggande programvara (middleware) utan att på ett djupare nivå förstå hur vi skulle konkret skulle kunna omsätta den tanken i praktiken. Kombinationen händelse och mellanliggande programvara gav ny fart åt den ursprungliga idén om en helt generell programvara, men i en helt annan skepnad än vad som ursprungligen var tänkt. Vår ursprungsidé hade landat i en helt generell mellanliggande programvara som med hjälp av händelser kan styra hur inlästa RFID-identiteter kopplas samman med en databas.

Så här i efterhand kan vi konstatera att vägen ifrån ursprungsidé till en generell mellanliggande programvara har varit krokig och tagit lång tid. Olika turer hit

och dit allt eftersom förståelsen ökat talar sitt tydliga språk, nämligen hur svårt systemutveckling är. En sak är säker, som det nämndes av flera av respondenterna, *"det är viktigt att sätta sig med branschfolket"* och att det tas hänsyn till användaren som *"bedömer om något är bra eller dåligt, även om det teoretiskt, tekniskt uppfyller alla önskvärda krav...tycker inte användaren att man uppfyller kraven blir det inte bra ändå"*. Samma sak uttryckt med andra ord är att *"man måste...så långt det går lära känna användarnas förväntningar och behov, det är det här klassiska misstaget man gör..."*. Systemutveckling är alltså svårt och en av de största utmaningarna är att lära känna användarens förväntningar och behov.

5.2 Översiktlig beskrivning av mellanliggande programvaran

Vår föreslagna lösning, RFID-Connect är ett program som automatiskt kopplar samman ett RFID-system med ett databasprogram. Genom att använda RFID-Connect kan den som har kunskaper om hur man skapar en tillämpning i ett vanligt databasprogram själv skapa sitt informationssystem. RFID-Connect utnyttjar fullt ut den funktionalitet som finns i respektive databasprogram, vilket innebär att man själv kan skapa informationssystem med omfattande funktionalitet. Databasprogrammet kan vara vilket program som helst som går att köra under exempelvis Windows XP med stöd för ODBC. Några begränsningar i tillämpningens storlek finns inte i RFID-Connect utan bestäms av databasens funktionalitet och kapacitet.

Exempel på användningsområden för RFID-Connect är:

- Uthyrningsverksamhet, intern och extern.
- Lagerhantering.
- Dokumenthantering.
- Inpasseringssystem.
- Timdebiteringssystem.

Det finns ingen begränsning till de nämnda exemplen, utan ett oändligt antal tillämpningar är möjliga.

RFID-Connect har ett grafiskt gränssnitt och är lätt att använda. Programmet är försett med en utvecklad hjälpfunktion. Vid drift av programmet är det fullständigt transparent vilket innebär att den enda kommunikation som sker med användaren är de funktioner som är definierade i databasprogrammet, exempelvis öppna formulär, utskrifter och meddelanden.

Eftersom de databaser som används stöder ODBC kan uppgifter från databaserna enkelt göras tillgängliga över Internet.

RFID-Connect är uppbyggt kring att hantera händelser. Genom Händelse-Hanteraren (figur 24) talar man om för systemet vilka yttre händelser, exempelvis att en RFID-tagg identifieras vid en läsare, som skall initiera händelser i sy-

stemet. På så sätt kan en tagg beroende på om den är känd eller okänd av systemet och beroende på vid vilken läsare den identifieras initiera en mängd olika skeenden. Det kan exempelvis vara att en för systemet okänd RFID-tagg kommer via läsare ”Kundkort”, programvaran är då inställd på att tolka detta som att en ny kund skall registreras. Händelsen initierar att ett formulär kommer upp på skärmen där relevanta kunduppgifter för en ny kund matas in i databasen. Vid nästa identifiering av samma tagg är den känd för systemet och aktiverar då ett annat skeende. Med andra ord ges användaren stor flexibilitet att styra vilka skeenden som ska inträffa beroende på kombinationen av läsare och tagg.

RFID-Connect består av två delar, en installationsdel och en servicedel.

5.2.1 Installationsdelen

Från installationsdelen av programmet görs nödvändiga inställningar och konfigurering av systemet. Från installationsdelen når man tre fönster som är centrala i arbetet att sätta upp systemet. Alla fönstren har likadan logisk och grafisk uppbyggnad:

1. ODBC-Hanteraren

För att systemet skall fungera krävs att man upprättar en ODBC-koppling till den databas man vill använda i systemet. I ODBC-hanteraren anges sökväg till databasen. I de fall den nås via ett nätverk anges dessutom ofta lösenord för åtkomst till såväl värddator som databas. ODBC-kopplingen namnges av användaren med ett valfritt namn som gör kopplingen lätt att identifiera. ODBC-Hanteraren tar alltså hand om alla nödvändiga inställningar i systemet, medan användaren därefter lätt kan ta bort, lägga till eller redigera ODBC-kopplingarna.

2. Läsar-Hanteraren

RFID-Connect möjliggör att ett godtyckligt antal RFID-läsare kan användas i systemet (figur25). I Läsar-Hanteraren namnges varje läsare med ett valfritt namn som gör den lätt att identifiera. I Läsar-Hanteraren visas också systemnamnet på respektive inkopplad läsare, som exempelvis kan vara en MAC-adress eller ett IP-nummer. Samma läsare kan kopplas mot flera tabeller och styra olika skeenden genom att användaren skapar skript för detta databasprogrammet. För tillfället kan dock varje tagg automatiskt endast generera ett skeende via varje läsare.

3. Händelse-Hanteraren

I Händelse-Hanteraren namnger användaren en händelse som gör den lätt att identifiera, exempelvis ”Ny kund” samt vilka villkor som skall uppfyllas vid den händelsen. För varje händelse anges dessutom vilken tabell och kolumn i databasen som händelsen skall arbeta mot. Tillgängliga alternativ listas upp automatiskt. Typiska åtgärder i programmet är att öppna formulär, köra makron och köra skript. På så sätt kan man koppla samman en kund med ett objekt, öppna ett formulär för inmatning i databasen, larma vid vissa händelser, styra

tillträde till lokaler, skriva ut fakturor, spåra objekt med mera bland ett mycket stort antal möjligheter.

Övriga inställningar

Åtkomsträttigheter och lösenord för ODBC-, Läsar-, och Händelsehanteraren, samt startinställningar sätts i samband med installation av programvaran. I samband med installation görs dessutom startinställningar för servicedelen. Vilket betyder att man anger om servicedelen alltid skall vara igång på datorn (autostart) eller startas manuellt.

5.2.2 Servicedelen

Servicedelen är den del av programmet som sköter det löpande arbetet med RFID-Connect. Den är helt transparent för användaren. När servicedelen är igång ”lyssnar” (bilaga 6) den hela tiden efter inkommande RFID-identiteter på datorns seriella portar och beroende på inställningarna i installationsdelen fattar den beslut om vilka åtgärder systemet skall vidta via händelsehanteraren.

5.3 Fördelning av uppgifter mellan användare och system

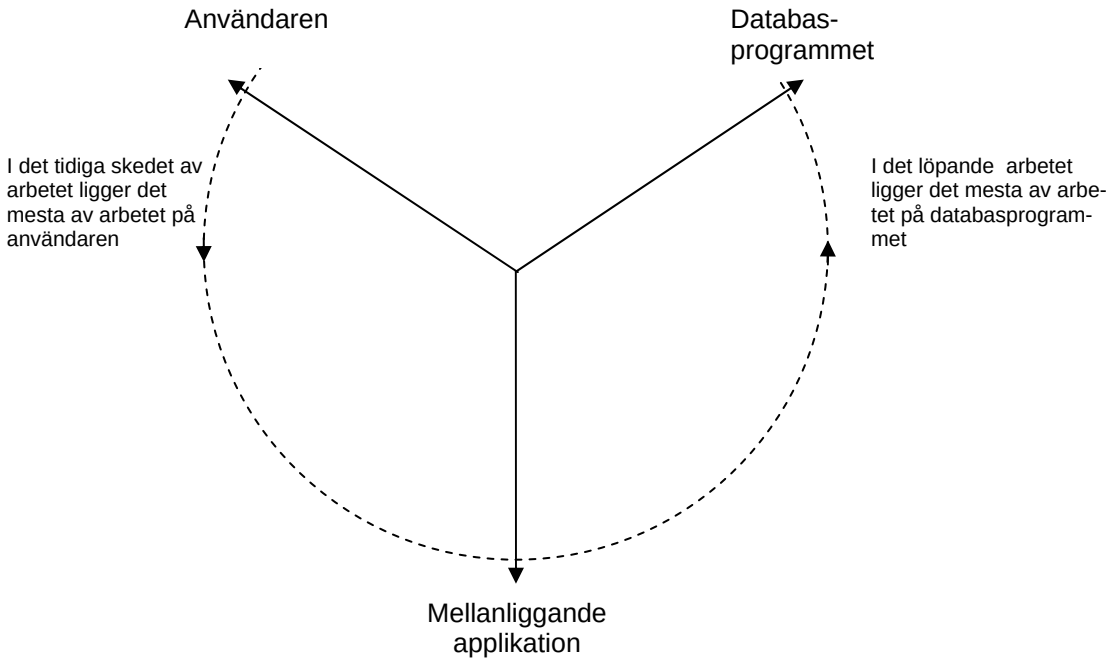
RFID-Connect som vi föreslår att det utformas, är en mellanliggande programvara som automatiserar installation, konfiguration och löpande körning av RFID-baserade informationssystem. I arbetet att realisera en sådan programvara finns ytterligare en fråga som hittills inte fått så mycket uppmärksamhet men som mycket av arbetet indirekt kretsat kring och den gäller hur uppgifterna inom systemet fördelas. En av fyra grundläggande karaktäristika för User-Centered Design som INUSE [2] och ISO 13407 för fram, är ”An appropriate allocation of function between user and system”. Det finns enkelt uttryckt tre aktörer i ett system som det vi beskriver och som kan vara med och dela på funktionerna:

1. Användaren.
2. Mellanliggande applikation.
3. Databasen.

Hela tanken med vårt arbete är att belasta användaren så lite som möjligt genom att automatisera uppgifter, både i installationsarbetet och i det löpande arbetet med informationssystemet. Inledningsvis konstaterar vi att frågan om arbetsfördelning kommer upp i två olika sammanhang, dels i uppbyggnaden av systemet, dels i användandet av systemet. När det gäller uppbyggnaden av systemet så innebär vårt upplägg med ett skal att vi lägger en stor börda på användaren att utforma databasen som är knuten till systemet. Naturligtvis kan det här arbetet överlåtas åt någon annan, men en bärande tanke är att någon kan utforma en databas med den funktionalitet man vill ha, men att den mellanliggande programvaran ger inte användaren något stöd i det arbetet. Det andra konstaterandet vi gör är att den mellanliggande programvaran i stor utsträckning automatiserar och konfigurerar systemet när databasen är klar. Det tredje konstaterandet är att när systemet väl är igång delar mellanliggande pro-

gramvara och databasprogrammet på hela arbetet. Den mellanliggande programvaran sköter genom händelsehanteraren vad som skall hända när en tagg identifieras av systemet. Databasprogrammet svarar därefter för det som användaren upplever, den faktiska funktionaliteten, som t ex att en förhyrning registreras, en faktura skrivs ut, en dörr öppnas eller någonting annat. Ett påpekande är att slutanvändaren aldrig kommer att uppleva den här fördelningen mellan aktörerna. Användaren kommer att utföra sina uppgifter som att lägga in nya kunder, registrera förhyrningar et cetera mot ett i övrigt sömlöst system där han eller hon inte upplever vem av de andra två aktörerna som gör vad. Det viktiga är att detta system utformats på ett sätt så att användaren kan få uppleva att informationssystemet uppfyller krav på att vara användarvänligt, funktionellt et cetera.

Fördelningen av arbetsuppgifter/-insatser mellan de tre aktörerna kan under uppbyggandet av systemet åskådliggöras med följande figur:



Figur 26. Fördelning av arbetsuppgifter/-insatser i systemet.

Den ursprungliga tanken med den mellanliggande programvaran var att den skulle innehålla funktionalitet utöver den vi nu föreslår. Programmet skulle då ta en större del av de uppgifter som vi nu lagt på databasprogrammet. Problemen med den designen blev snart tydliga för oss, den skulle utan tvekan innebära att vi fick överge tanken på en generell lösning för många typer av tillämpningar. Den vägen vi valt, med generella möjligheter, innebär att vi får lämna användaren utan stöd i arbetet att bygga upp databasen. Alternativet att ge användaren stöd i utformandet av databasen hade inneburit att den mellanliggande programvaran fått specialiseras för vissa tillämpningar, eller fått utformas mycket komplex för att kunna hantera så många tillämpningar som

möjligt. Vår uppfattning här är att kunnandet om hur man bygger databaser är betydligt mera spritt än kunnandet hur man konfigurerar RFID-system, och att det således är för konfigurering av RFID-systemet stödet behövs. Att sedan databasen byggs upp för datafångst genom RFID spelar i sammanhanget mindre roll. Sådant man särskilt måste ta hänsyn till för att det är ett RFID-system, är exempelvis att ett nyckelfält bestäms av RFID-taggarnas identiteter. Samt att tabeller och funktioner får byggas upp efter de förutsättningar som ges av läsare och taggar. I övrigt utformas allt i databasprogrammet som vanligt.

5.4 Småskalighet

Vår inriktning från början av arbetet har varit att göra en programvara för småskaliga tillämpningar. Som vi sett under arbetets gång har inriktningen i vårt arbete gått från att göra en programvara med väsentligt mer egen funktionalitet till den skalmodell vi nu föreslår, en lösning som så långt som möjligt drar nytta av och utnyttjar databasprogrammets funktioner. Att vi rör oss mot ett skal innebär att de tankar vi haft om småskalighet delvis kommer i en annan dager. Det finns under förutsättningar att databasprogrammet klarar av uppgiften ingenting i RFID-Connect som begränsar användandet till enbart småskaliga tillämpningar. Som vi sett tidigare (kapitel 2.6) är begreppet småskalighet något man snarare definierar med ord än mäter. Vi förde i teoridelen fram tankar kring antalet utvecklare, behov av formella tekniker, förhållningssätt vid skapandet av programvaran et cetera. Ett som vi tror viktigt kriterium i sammanhanget är vilken grad av specialisering man önskar av systemet, där vi menar att mindre grad av specialisering indikerar småskalig användning och omvänt. Med grad av specialisering avser vi i vilken utsträckning tillämpningen klarar sig med den funktionalitet som finns i databasprogrammet. Hög grad av specialisering i det här perspektivet betyder således att tillämpningen behöver mer funktionalitet än databasprogrammet, med eller utan för mycket skript och makron kan leverera. Exempelvis kan det vara möjligheter att effektivt arbeta ihop med ett omfattande redovisningssystem, möjlighet att arbeta i miljöer med komplexa sammansättningar av operativsystem, krav på särskilda buffringar av många och mycket täta datafångster, krav på att samtidigt kunna arbeta mot flera databaser et cetera. Låg grad av specialisering innebär att nödvändig funktionalitet finns klar att använda i databasprogrammet.

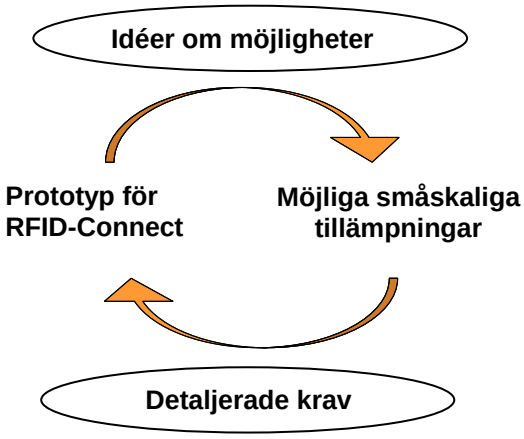
Sätter vi in det synsättet i den här diskussionen menar vi att det är användarnas krav på specialisering som slutligen avgör om man skall satsa på en lösning grundad på standardprogramvaror eller en från början specialanpassad lösning. Med behov och önskemål om allt mer specialiserade funktioner kommer den tänkte användaren att bli allt mer benägen att lämna den trots allt begränsade funktionalitet som finns i enklare databasprogram till förmån för en specialanpassad lösning. Även om det är teoretiskt möjligt att uppnå önskad funktionalitet med standardfunktioner, makron och skript riskerar vi att få det som en av respondenterna uttryckte som ” *systemet måste vara enkelt från början så att man inte lappar och lagar för mycket under resans gång*”.

Frågan om när man skall använda en mellanliggande programvara som den vi föreslår eller en specialanpassad lösning menar vi kan besvaras genom att utgå från begreppen i Quality in Use. För att kraven för Quality in Use skall kunna uppfyllas bör underliggande nivåer också uppfylla dessa krav. När den tillämpning man vill skapa, det vill säga mellanliggande programvara och databasprogrammet med sin funktionalitet, tillsammans inte uppnår förutsättningar för QiU, bör man överväga en från början specialanpassad lösning. Skälet till detta är naturligtvis att tillämpningen annars inte har förutsättningar att fungera med de användare och i den miljö den är tänkt att verka.

5.5 Realisering

Vårt arbete syftade till att undersöka hur en generell mellanliggande programvara kunde utformas för att underlätta tillkomsten av småskaliga RFID-tillämpningar. Resultatet av arbetet skulle leda till en beskrivning av en sådan mellanliggande programvara. I kapitlet om avgränsningar har vi deklarerat att vi inte har för avsikt att realisera programvaran. När man beskriver en programvara kan det ske på många sätt, vi har exempelvis sett hur Sommerville (2001) resonerar kring krav på olika nivåer, å ena sidan användarkrav med en relativt låg grad av specificering och å andra sidan systemkrav som är detaljerade. I det här perspektivet blir resultatet av vårt arbete snarast en beskrivning av användarkraven. En beskrivning i prosa där mycket arbete återstår innan en realisering. Naturligtvis är det lockande att reflektera över vilka moment som återstår, inte minst med tanke på att mycket av det arbete som hittills är gjort är en del i en realisering.

Vi har tidigare i kapitel 2 beskrivit hur prototypen itereras för att kunna möta Quality in Use. Vi använde då figur 10 för att beskriva hur den mellanliggande programvaran både skapar möjligheter och underställs krav. Så länge programvaran inte är färdig ser bilden annorlunda ut. I stället för nya möjligheter får vi prata om idéer eller tankar kring nya möjligheter. Prototypen och idéer om nya möjligheter leder till möjliga småskaliga tillämpningar. Vi har med denna utgångspunkt genomfört tre intervjuer med domänexperter och dessa har levererat tillbaka en uppsättning användarkrav på systemet belysta genom terminologin i QiU. För att kunna realisera den mellanliggande programvaran menar vi att vi även behöver ha en rad detaljerade krav definierade. Vi vill med andra ord utifrån möjliga småskaliga tillämpningar få tillbaka detaljerade krav för att realisera programvaran. De detaljerade kraven kan exempelvis röra sig om behov av buffring när man måste mellanlagra RFID-identiteter i avvaktan på att databasprogrammet skall hinna åtgärda dem, säkerhetskrav, krav på att använda speciell hårdvara et cetera.



Figur 27. CERES-glasögonen för en icke färdig mellanliggande programvara. Figuren är delvis återgiven och modifierad utifrån CERES-glasögonen (Svensson et al, 2004, s. 7; Svensson, 2004).

Vi har tidigare konstaterat att de tänkta användarna, i den mån man överhuvudtaget kan finna dem, sannolikt bara har vaga begrepp om sina eventuella krav på den tänkta programvaran. Mot detta talar den skarpa kontrasten i hur man bedriver användarcentrerad systemutveckling, där just användarnas syn på systemet betecknas som mycket viktig. Vi måste med andra ord, hitta sätt att finna de detaljerade kraven på den mellanliggande programvaran för det fortsatta utvecklingsarbetet, helst ihop med slutanvändare. Vi vill föreslå i huvudsak två sätt att göra detta:

1. Söka upp och intervjua personer som idag har informationssystem som antingen är RFID-baserade eller inte. I den utsträckning det går att finna småskaliga lösningar är detta givetvis önskvärt. Systemen bör finnas inom de områden som är intressanta för oss, det kan gälla uthyrningsverksamhet, system för timdebitering, enklare lagerhanteringssystem et cetera. Systemägarna får då intervjuas kring vilka tjänster de har respektive vill ha. Utifrån dessa önskemål om tjänster får systemet beskrivas på en mer detaljerad nivå.

2. Fortsätta med djupare intervjuer av domänexperterna. Dessa har hittills fått tjäna som ett slags ombud för den anonyma användargruppen. Vi är helt övertygade om att det bland dessa finns mycket detaljerade kunskaper kring den här typen av system. Inte minst har man byggt specialanpassade system för kunder och har samlat på sig erfarenheter kring detta.

Om vi ser det fortsatta arbetet utifrån QiU så handlar det nu om att realisera ett informationssystem, det vill säga de två innersta cirklarna enligt figur 11. Om systemet i sin helhet uppfyller kraven för external quality finns förutsättningar för att systemet i sin helhet, dvs med sina slutanvändare och i sin miljö kan fungera väl, dvs vi uppnår Quality in Use. För det fortsatta arbetet med de detaljerade kraven oavsett om vi arbetar med slutanvändare eller domänexperter, får vi således fortsätta med perspektiven på functionality, usability osv.

5.6 Att underlätta implementeringen

Att göra slutanvändaren medvetenheten om lämpliga användningsområden är viktigt med tanke på att denna måste känna till vad RFID-tekniken kan användas till. Vi tror att goda exempel underlättar för slutanvändaren att se tillämpningar i sin egen verksamhet och omgivning. Under intervjuerna framgick många intressanta användningsområden som skulle vara mycket lämpliga för småskaliga RFID-system, se avsnitt 5.1. Ett hinder för att dessa småskaliga RFID-system ska bli verklighet är att kostnaderna för att utveckla informationssystem kan bli mycket stora. Av intervjuerna framgick det att *"hårdvaran är enkel att köpa in, men det är lätt att lägga det femdubbla beloppet på att utveckla programvaran..."*. Ett annat hinder är att RFID-tekniken är svår för användarna att implementera på egen hand. En av respondenterna klargjorde att *"implementering skulle kunna underlättas av ett sådant här system eftersom användarna idag som regel behöver hjälp att få det att fungera"*. Om vi kan hjälpa slutanvändaren med att klara av dessa hinder kommer säkerligen fler tillämpningar av småskaliga RFID-baserade system att se dagens ljus.
Istället för att titta på olika användningsområden för att underlätta för slutanvändarna att se behovet av RFID-tekniken, kan ett annat sätt vara att undersöka vilka grundläggande funktioner som vanligen efterfrågas i denna typ av system. Under intervjuerna framgick det att de grundläggande funktionerna som ofta efterfrågas är:

- Att identifiera något.
- Att leta efter något.
- Att se var något befinner sig.
- Att räkna något.
- Att registrera något.
- Att associera något med något annat.
- Att göra data åtkomliga.

Dessa grundläggande funktioner är av en mycket generell karaktär och det finns därför inga begränsningar av möjligheterna att finna tillämpningar i slutanvändarens verksamhet. Det är slutanvändaren som bäst känner sin verksamhet och därmed bäst kan se var lönsamheten av en tillämpning av RFID-tekniken är som störst. Vi tror att även dessa grundläggande funktioner kan bidra till att hjälpa användaren att se behovet av RFID-tekniken, och hur denna kan hjälpa till med för att underlätta i slutanvändarens verksamhet. Ser slutanvändaren behovet av RFID-tekniken, behövs hjälp för att få RFID-systemet att fungera, således fyller en mellanliggande programvara som RFID-Connect sitt syfte.

En generell mellanliggande programvara som hjälper slutanvändaren att koppla ihop RFID-tekniken med en databas tror vi därför kommer att underlätta för implementeringen av småskaliga RFID-system. Detta beror på att användaren kan göra mycket av arbetet med applikationen på egen hand, direkt i databasen, och på så vis är kostnaderna för utveckling av användarens programvara

inte längre ett lika stort hinder. Det beror även på att en mellanliggande generell programvara minskar svårigheterna att koppla ihop RFID-teknikens inlästa RFID-identiteter med rätt association i en databas. Att koppla inläst identitet belyste en respondent med de tänkvärda orden *"ett nummer säger ingenting om det inte kopplas till något"*.

5.7 Quality in use

Quality in use eller användarkvalitet, är ett övergripande begrepp som kan sägas vara ett kvalitetsmått på en programvara när den används. ISO 8402 (1994) definierar kvalitet som *"alla sammantagna egenskaper hos ett objekt eller företeelse som ger dess förmåga att tillfredsställa uttalade eller underförstådda behov"*. I samband med beaktande av QiU syftar behovet tillbaka på användarens behov när denne använder programvaran.

Quality in use är uppbyggd i tre olika nivåer, intern kvalitet, extern kvalitet och quality in use. Som vi tidigare förklarat i samband med figur 12 är intern kvalitet något som påverkar extern kvalitet, medan extern kvalitet i sin tur påverkar quality in use. Quality in use ställer i sin tur krav på extern kvalitet och extern kvalitet ställer i sin tur krav på intern kvalitet. Intern och extern kvalitet kan även sägas beskriva programvarans kvalitet medan quality in use beskriver effekterna när programvaran används. Extern kvalitet kan även sägas vara kvalitet på det vis som användaren uppfattar och beskriver den, medan intern kvalitet istället är kvalitet på det vis som systemutvecklaren uppfattar och beskriver den.

I systemutvecklingen av en generell mellanliggande programvara har vi under intervjuerna huvudsakligen samlat in material kring extern kvalitet genom de 6 begreppen som de presenterats i ISO/IEC 9126-1 (1998). Begreppen är functionality, reliability, portability, usability, effeciency och maintainability. Kopplat till figur 12, som beaktar kvalitet i dess livscykel, räcker den externa kvaliteten inte till för att kunna beakta quality in use. Det saknas således även ett beaktande av den intern kvalitet för den mellanliggande programvaran, så som systemutvecklaren ser behoven. Om vi utgår ifrån figur 11 så har vi varken användare eller omgivning eftersom den mellanliggande programvaran i detta skede varken realiserats eller implementerats. Det är viktigt att förstå att både intern och extern kvalitet påverkar quality in use, således behövs alla dessa delar för att programvaran som helhet ska fungera bra under användningen.

Vi redovisar de synpunkter som respondenterna redogjort, för kring de 6 begrepp som beskriver den externa kvaliteten på en övergripande nivå. Anledningen till detta är att det är svårt att säga något om vad användarna egentligen ser för tillämpningar i samband med RFID-tekniken och i vilken miljö dessa kommer att användas. Det vi även kan konstatera utifrån vår undersökning är att resultatet av undersökningen har förskjutits mot en generell mellanliggande programvara allt eftersom vår förståelse blivit större. Användarnas behov har med andra ord förskjutits från att omfatta rena tillämpningar, t ex branschspeci-

fikt, till att istället omfatta en generell mellanliggande programvara. Detta har inneburit att det som blivit resultatet av undersökningen snarare har hjälpt oss att hitta omfattningen på det som går att göra generellt snarare än att beskriva den externa kvaliteten i detalj. Vidare kan vi även konstatera vad som anses mer viktigt och mindre viktigt att beakta beträffande den externa kvaliteten.

Functionality
Funktionalitet är en självklarhet och viktades som mycket viktig av samtliga respondenter, som någon av dem konstaterade *"allt annat faller om inte det fungerar"*. Funktionalitet beror på vad användaren förväntar sig av systemet, bland annat nämndes att *"användarna förväntar sig funktionalitet"* . Vad användarna förväntar sig går inte att svara på eftersom tillämpningarna inte är gjorda ännu, men att det är mycket viktigt råder det inga tvivel om.

Reliability
Reliability har ej framstått som särskilt relevant att beakta för denna typ av småskaliga RFID-applikationer. Respondenterna bedömer att reliability har liten eller stor betydelse beroende på vilken typ av tillämpning som det är tal om. Som en av dem nämner *"t ex går det ej att jämföra ett styrsystem i ett flygplan med ett tvätteri där du tvättar kläder. Ett flygplan kan komma att störta om inte systemet svarar inom 15 millisekunder medan det i tvätteriet har föga betydelse"*.

Portability
Här har respondenterna haft relativt olika uppfattning om vad som är och inte är viktigt. Någon menar att det ej alls är relevant eftersom installation sker med hjälp av en CD-skiva för PC, medan alla har lyft fram att portabla läsare är mycket viktigt för att exempelvis kunna leta efter saker, i synnerhet om de är stora och tunga. En annan viktig aspekt var att det kan finnas olika täckningsområden för en handburen läsare. Detta innebär en blandning av online och offline system som anses vara av stor betydelse i samband med portability. Med tanke på svaren bedömer vi att portability har stor betydelse.

Usability
Usability är lika självklart som functionality och anses ha en mycket stor betydelse enligt alla respondenterna. Några av kommentarer var att *"allt annat är avhängigt usability"* och systemet bör vara *"näst intill självinstruerande"*. Detta ansvar vilar till stor del på användaren när dennes tillämpning tar form. Vi kan dock säkerställa att gränsnittet på den mellanliggande programvaran är lätta att använda.

Effeciency
Även här finns blandande synpunkter bland respondenterna. En av dem påpekar att *"annars så köper kunden inte systemet"* medan en annan förklarar att det i detta fallet inte är tal om ett prestandasystem. Det framgår av resultatet att det är stor skillnad på ett stort och ett småskaligt system. Vi bedömer att effeciency är av liten till stor betydelse beroende på tillämpningens omfattning.

Maintainability
Maintainability tycker respondenterna är av stor betydelse. De menar att programvaran måste gå att underhålla och att *"annars är det lätt att den förfaller"*. Samtidigt påpekade någon av dem att själva hårdvaran byts snarare ut istället för att underhållas.

En sak som vi slutligen också måste säga är att processkvalitet är något som har ett starkt samband med produktkvalitet (Sommerville, 2001; Håkansson, 2000). Även processkvaliteten behöver beaktas för att underlätta systemutvecklingen av en programvara med rätt kvalitet. För att säkerställa processkvaliteten har vi använt oss av INUSE [2].

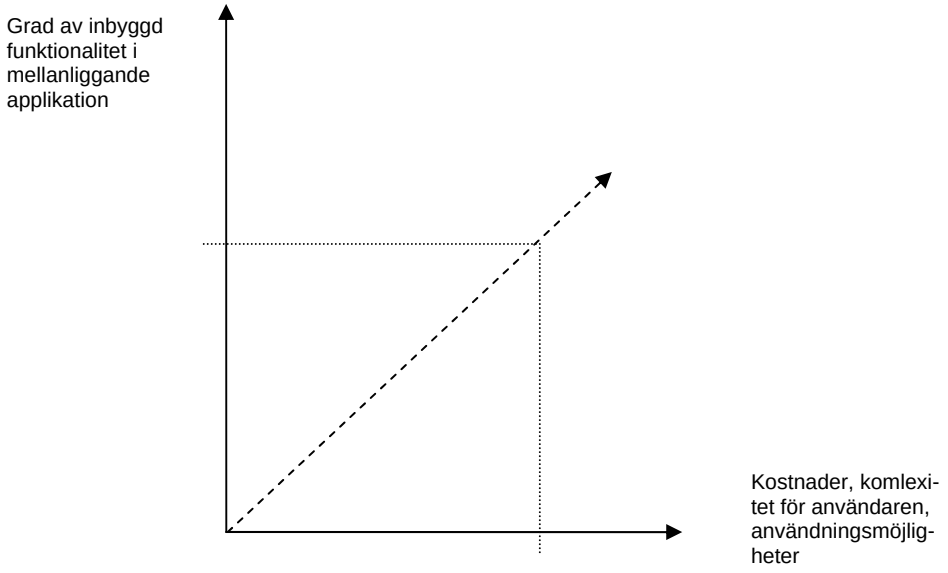
Vårt bidrag ligger i linje med det som föreskrivs i Quality in use och vi har använt oss av INUSE [2] för att säkerställa en bra processkvalitet under systemutvecklingen. För att designa den slutliga versionen av de tillämpningar som framledes kommer att realiseras och implementeras behöver användaren bidra med de delar som återstår för att även helheten ska bli bra.

5.8 Vidare forskning och arbeten
Det finns väldigt många olika projekt som är möjliga att genomföra med RFID-teknik, så många att det upplevs som lite av ett problem att hinna med att bearbeta alla idéer. En del av problematiken är att det är dyrt att utveckla programvarorna som ofta behövs för det ska bli någonting av det hela. Det finns helt klart behov av kartlägga hur utvecklingen av dessa idéer bättre skulle kunna tas om hand för att på så vis kunna främja utvecklingen av de olika RFID-baserade tillämpningar som annars inte skulle se dagens ljus.

En annan intressant fråga gäller hur man sätter rätt pris på information, dvs hur man värderar den för att kunna avgöra om ett informationssystem grundat på RFID är en ekonomiskt försvarbar lösning eller ej. Nu har vi inte undersökt om detta redan är gjort, en möjligen ny parameter i en sådan värderingsmodell borde i så fall vara möjligheten att få informationen snabbt, samt möjligheter att arbeta även mot globala databaser.

När det gäller komplexiteten i en mellanliggande programvara vill vi också för kommande arbeten skicka med en fråga om risken att programvaran för att täcka upp så många användningsområden och specialiseringar som möjligt, blir alltför komplex för att uppfylla sitt syfte. Våra tankar här är att en programvara som designas för att förenkla och automatisera småskaliga tillämpningar måste hållas enkel. I våra intervjuer med domänexperterna förstod vi snabbt att önskemålen om funktioner från användarnas sida snabbt kan bli mycket omfattande. Om dessa funktioner läggs i en programvara som skall kunna användas till en rad olika slag av tillämpningar innebär det att antalet funktioner blir stort och att antalet funktioner som man inte använder sannolikt också blir stort. Att reda ut vilka funktioner som skall finnas blir också ett mycket stort och grannlaga arbete. Vår lösning på detta är ju som bekant att erbjuda de funktioner som

databasprogrammet medger, räcker inte det får användaren lösa sin tillämpning med en specialanpassad lösning. I den avslutande diskussionen har vi resonerat utifrån ett antagande om att graden av funktionalitet i en mellanliggande programvara är direkt proportionerligt mot kostnaden, komplexiteten och användningsmöjligheterna hos programvaran. Vid någon nivå blir applikationen för dyr och för komplex för att använda och då hjälper det inte att användningsmöjligheterna är stora. Om antagandet stämmer kan vi argumentera för att den strategi vi valt innebär låga kostnader och låg komplexitet men också små användningsmöjligheter. Den sista besvärande omständigheten hanterar vi som bekant genom att låta databasprogrammet svara för funktionaliteten.



Figur 28. Förhållande mellan inbyggd funktionalitet, kostnader, komplexitet för användaren och användningsmöjligheter.

5.9 Epilog

Vårt arbete startade med en mängd frågor kring RFID och informationssystem. För att reda ut den som vi bedömde viktigaste frågan har vi fördjupat oss i inbyggda system, databaser, ODBC, RFID, Quality in Use, prototyping, hur man hanterar kravprocessen och småskaliga tillämpningar. Vi har skapat referensapplikationer, prototypat och intervjuat. Som om inte detta vore nog har vi fördjupat oss i ytterligare frågor som visade sig vara återvändsgränder. Redan tidigt förstod vi att vi hade att göra med en fråga av delvis explorativt slag. Ofta under arbetets gång har vi upplevt att vi jagat ett mål som inte riktigt velat stå stilla. I allt detta arbete har vi haft roligt och vi har tveklöst fört frågan framåt. Vi är också ganska övertygade om att vi bidragit till att fixera målet. Vi hoppas att någon vill ta upp tråden och fortsätta vårt arbete. Vi kan lova ett roligt område som definitivt har framtiden för sig. Med visst vemod stänger vi här, fullt förvissade om att det är nu det riktigt roliga börjar...

"When the focus shifts from information systems to information services, there is a need for a new, more application oriented discipline, developing innovative services for the nomads of information society".

(Dahlbom, 2001, s. 9)

6 Referenser

6.1 Artiklar/konferensrapporter/böcker

Apelkrans, M. & Åbom, C., (2001). *OOS/UML - En objektorienterad systemutvecklingsmodell.* Lund: Studentlitteratur.

Andersen, E.S. (1994). *Systemutveckling – principer, metoder och tekniker.* Lund: Studentlitteratur.

Asplund, J. (1970). *Om undran inför samhället*. Lund: Argos.

AUTO-ID Center , Technology Guide (2002). Massachusetts institute of Technology.

Backman, J. (2000). *Rapporter och uppsatser.* Lund: Studentlitteratur.

Beekman, G., Rathswohl, E, J. (2001). *Computer Confluence, Business Edition, Exploring Tomorrows Technology.* Upper Saddle River: Prentice Hall.

Bevan, N. (1999). Quality in use: Meting user needs for quality. *The Journal of Systems and Software 49*, 89-96.

Borgström, H. (2004). Bättre kontroll med kretsmärkta varor. *Forskning & Framsteg,* nr 4, 2004.

Bray, I. K., (2002). *An introduction to requirements engineering.* Harlow: Addison-Wesley.

Carlsson, B. (1990). *Grundläggande forskningsmetodik för medicin och beteendevetenskap*. Göteborg: Almqvist & Wiksell.

Conolly, T., Begg, C. (2002). *Database systems - A practical approach to design, implementation and management (3rd ed.).* Harlow: Addison Wesley.

Dahlbom, B. (2001). *From Systems to Services.* Göteborgs universitet, Institutionen för Informatik.

Date, C.J. (2004). *An Introduction to Database Systems (8th ed.)*. USA: Addison-Wesley.

Elmasri, R., Navethe, S. B., (2004). *The fundamentals of database systems (4th ed.).* Boston: Addison-Wesley.

Emory, C.W., & Cooper, D.R. (1991). *Business Research Methods.* Boston: Irwin.

Fildes, J. (2002). Chips with everything. *New Scientist*, *October*, 45-47.

Finkenzeller, K. (1999). *RFID Handbook.* New York: Wiley.

Gyger, T., Desjeux, O. (2001). EasyRide: Active Transponders for a Fare Collection System. *IEEE Micro, November-December 2001, 36-42.*

Halvorsen, K. (1992). *Samhällsvetenskaplig metod.* Lund: Studentlitteratur.

Hawryszkiewycz, I., (2001). *Introduction to systems analysis & design (5th ed.).* Frenchs Forest: Prentice Hall.

Heiat, A., & Heiat, N. (1997). A Model For Estimating Efforts Required For Developing Small-Scale Business Applications. *Systems Software, 39*, 7-14.

Holme, I.M., & Krohn Solvang, B. (2001). *Forskningsmetodik Om kvalitativa och kvantitativa metoder.* Lund: Studentlitteratur.

Howes, R., Williams, A., & Evans, M. (1999). A read/write tag for low cost applications. *The institution of Electrical Engineers, Savoy Place, London.*

Håkansson, B. (2000). *Egendeklaration av programvaror Svensk metod för kvalitetsmärkning av programvaror med internationella standarder som grund.* Stockholm: Via Teldok.

Kvale, S. (1997). *Den kvalitativa forskningsintervjun.* Lund: Studentlitteratur.

Laitinen, M. (2000). Scaling Down Is Hard to Do. *IEEE Software, September/October*, 78-80.

Lundahl, U. & Skärvad, P-H (1999), *Utredningsmetodik för samhällsvetare och ekonomer*. Lund: Studentlitteratur.

Löwgren, J., Stolterman, E. (2001). *Design av informationsteknik.* Lund: Studentlitteratur.

Maciaszek, L. A., (2001). *Requirements analysis and system design Developing information systems with UML.* Harlow: Addison-Wesley.

Ottosson, S. (1993). *Lönsam innovationsverksamhet. Från idé till medelstort företag.* Floda: Onix venture AB.

Patel, R. & Tebelius, U. (red) (1987) *Grundbok i forskningsmetodik*. Lund: Studentlitteratur.

Raza, N., Bradshaw, V., & Hague, M. (1999). Application of RFID Technology. *IEEE Collocium.* 1-5.

Ryberg, J. (2004). Jättar kräver ständig koll på varan. *Ny teknik*, 28 jan 2004.

Sakamura, K. (2001). Radio Frequency Identification and Noncontact Smart Cards. *IEEE Micro, November-December 2001, 4-6.*

Skansholm, J. (2000). *Java direkt med sving.* Lund: Studentlitteratur.

Sommerville, I. (2001). *Software engineering (6th ed.).* Harlow: Addison-Wesley.

Stanford, V. (2003). Pervasive Computing Goes the Last Hundred Feet with RFID Systems. *Pervasive computing Vol(2), April-June 2003*, 9-14.

Starrin, B., Larsson, G., Dahlgren, L. & Styrborn, S. (1991), *Från upptäckt till presentation*. Lund: Studentlitteratur.

Ström, P. (2002). *Prylarna snackar. Maskin till maskinkommunikation (M2M), telematik och "ubiquitous Internet".* Uppsala: Publishing House AB.

Svensson, B., Jonsson, M., Larsson, T., Bilstrup, U., Gaspes, V., Wiberg P., Åhlander, A., Hammerstrom, D., Vasell, J. (2004). *Cooperating Embedded Systems, A Proposal to the Knowledge Foundation*. Halmstad: CERES.

Takaragi, K., Usami, M., Imura, R., Itsuki, R., & Satoh, T. (2001). An ultra small individual recognition security chip. *MICRO IEEE, 21,* (6), 43-49.

Tuttle, J.R. (1997). Traditional and emerging technologies and applications in the Radio Frequency Identification (RFID) industry. *IEEE Radio Frequency Integrated Circuits Symposium, 1-2*, 5-8.
Ullman, J.D., & Widom, J. (2002). *A first course in database systems (2nd ed.).* Upper Saddle River: Prentice Hall.

Wiedersheim-Paul, F. & Eriksson, L.T. (1999). *Att utreda, forska och rapportera*. Malmö: Liber.

Wolf, W. (2002). What is embedded computing? *Computer, January 2002, 136-137.*

6.2 Personlig kommunikation

Laadoe, B. (2003). Presschef, IBM. Kontaktad på telefon 27/1. Vi har tagit del av intern dokumentation över Vasaloppets tidtagningssystem.

Svensson, B. (2004). Professor, projektledare för CERES. Intervjuad 21/4 kring inbyggda system och CERES-modellen.

6.3 Standarder

ISO 8402, (1994). *Quality management and quality assurance -Vocabulary.*
ISO/IEC 9126-1, (1998). *Software product quality – Part 1: Quality model*.
ISO 9241-11, (1998). *Guidance on usability specification and measures*.

6.4 Uppsatser

Ahlkvist Scarfeld, T. (2001). *An analysis of the Fundamental Constraints on Low Cost Pas-sive Radio-Frequency Identification System design. (In partial Fulfillment of the Require-ments for the Degree of Masters of Science)*. Massachusetts Institute of Technology. De-partment of Mechanical engineering.

Johansson, J., Granholm, J., Berggren, M., & Pihl, H. (2002). *Projektrapport WPS (Wireless Parking Solution) (Projekt på ICT-programmet)*. Högskolan i Halmstad, Sektionen för Informationsvetenskap, data- och elektroteknik, 301 18 Halmstad.

6.5 Webbdokument

[1] Assalub AB (2003). URL: http://www.tekniq.nu/ref_artiklar/assalub/, 2003-07-25.

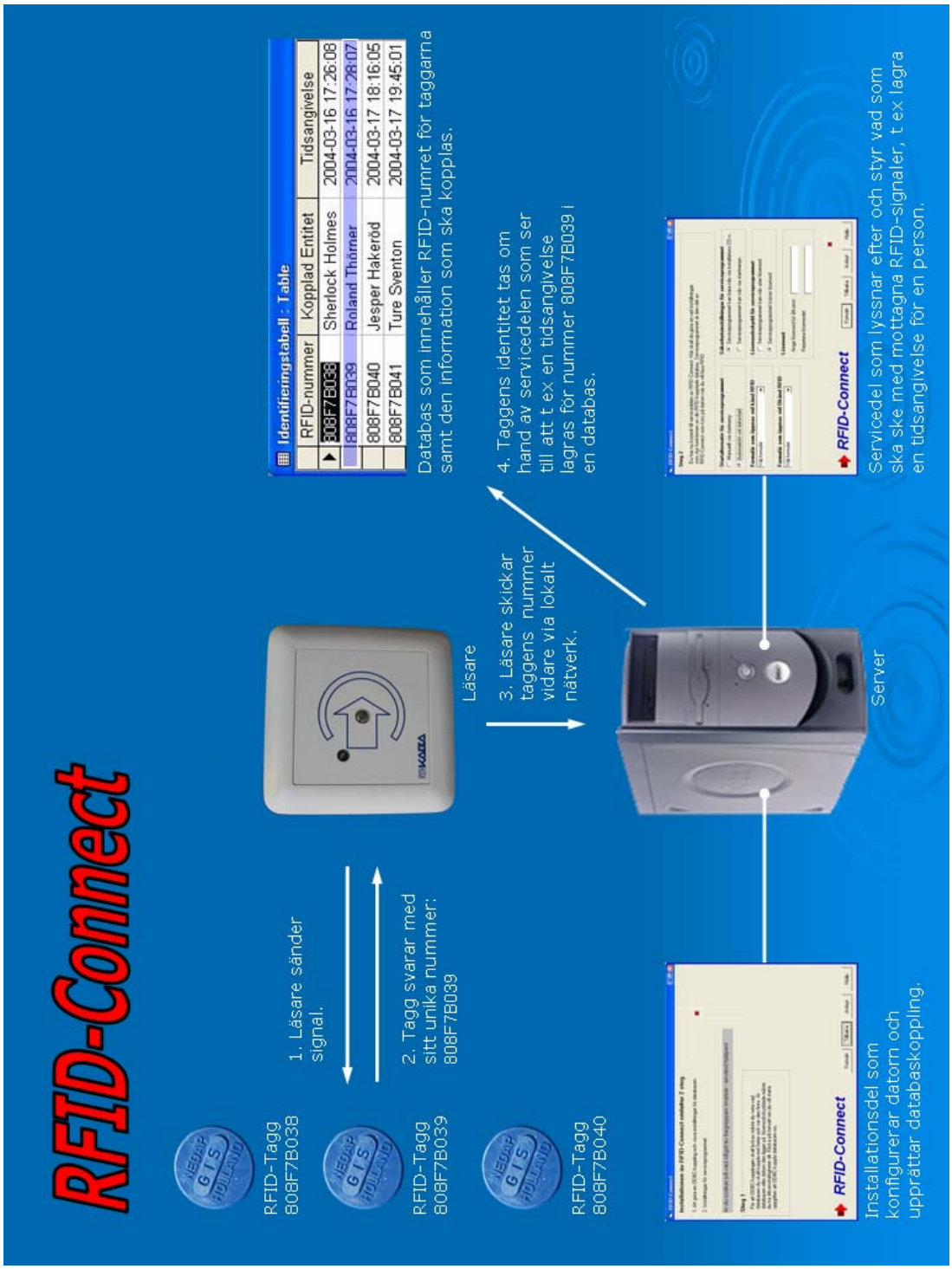
[2] INUSE 6.2 (2003). URL: http://www.ejeisa.com/nectar/inuse/6.2/summary.htm, 2003-12-25.

[3] Mindescape (2004) URL: http://www.mindescape.se, 2004-04-29

[4] TeknIQ-projektet (2003). URL: http://www.tekniq.nu, 2003-01-30.

[5] Vasaloppet (2003) URL: http://www.vasaloppet.se/Tavlingar/Resultat/index.html, 2003-01-26.

Bilaga 1, Översiktsskiss



Bilaga 2, Dokumentation av systemutveckling

RFID-Connect Systemutvecklingsdokumentation

Upprättad av Roland Thörner
INUSE 6.2/1.5.1

Understand and specify the context of use

The characteristics of the intended users

RFID-Connect är en generisk programvara som inte vänder sig till en känd grupp användare. För utveckling av programvaran får vi därför göra vissa mer eller mindre välgrundade antaganden. Vi vänder oss till en kundkrets som vill skapa egna RFID-tillämpningar, antingen för att implementera egna taggar i ett sammanhang eller för att läsa taggar som någon annan har implementerat. Mot denna bakgrund får vi förutsätta att kunden har tekniska kunskaper som ligger relativt högt. Vidare förutsätter bruket av RFID-Connect att det finns en fungerande databas att köra emot. När det gäller bruket av RFID-Connect kan användarna förenklat delas upp i två grupper, implementationsgruppen och en grupp ”vanliga användare”. För gruppen vanliga användare skall bruket av RFID-Connect inte innebära några som helst kunskapskrav utöver att hantera databasprogrammet, den användningen skall med andra ord vara helt transparent. När det gäller implementations- gruppen, den grupp av användarna som skall göra den inledande installationen och konfigurationen av systemet är kunskapskraven högre. Vi kan också här anta att det i den här gruppen finns viss IT-vana, dock måste vi även planera för det motsatta – att kunskaper till stor del saknas. Med tanke på att gruppen är heterogen och okänd måste dock programvaran förses med omfattande hjälpsystem för implementationsgruppen.

The task the users will perform

RFID är en teknik som möjliggör snabb och säker identifiering av entiteter. RFID-Connect hjälper användaren att göra kopplingar till en databas så att användningsområden och möjligheter med tekniken ökar avsevärt. Systemets mål är att koppla samman RFID med en databas så att användaren får en helt transparent upplevelse av RFID-Connect, det enda den vanlige användaren skall uppleva av systemet är gränssnittet mot databasprogrammet. Att RFID är en teknik med hög säkerhet innebär att RFID-Connect skall stötta användaren så att hans eller hennes insats blir lika säker som tekniken i övrigt. Med tanke på att RFID ofta, men inte alltid, används i sammanhang där många entiteter skall hanteras skall programvaran stötta användaren i att arbeta snabbt med systemet.

När det gäller implementation och underhåll av systemet ställs större krav på användarna. Typiska uppgifter här är att lägga till, ändra och ta bort entiteter, att fatta beslut om och utforma gränssnitt kring vad systemet skall göra när en okänd entitet identifieras et cetera. En del av dessa uppgifter ligger i databasprogrammet i systemet andra i RFID-Connect.

The environment in which the users will use the system

Ett system som RFID-Connect kan antas användas i situationer där användaren har behov av snabbt och säkert kunna identifiera en entitet. Typiska miljöer är kundmottagningar och butiker. I vissa fall finns ingen användare i traditionell mening, vid exempelvis ett inpasseringssystem blir bäraren av RFID-taggen användare, ett förhållande som gör transparensen än mer angelägen.

Specify the user and organizational requirements for the system

Range of relevant users and other personell in the design

Användargruppen är okänd, systemet utvecklas av två personer. Till utvecklingsgruppen kopplas ett antal experter inom olika områden. Programvaran som skapas är för småskalig tillämpning, dock säger detta inget om antalet användare.

Clear statements of design goals

Itereras fram under utvecklingen

An indication of appropriate priorities for the different requirements
Bland QIU kriterierna har följande prioritering gjorts:

1. Functionality
2. Usability
3. Maintaiability

Provision of measurable benchmarks against which the emerging design can be tested
Situationsspecifikt, får i så fall tas fram kriterier inom varje användningsområde

Evidence of acceptance of the recuirements by the stakeholders or their representtatives
Situationsspecifikt

Acknowledgement of any statutory or legislative requirements, for example for health and safety
Situationsspecifikt

RFID-Connect Systemutvecklingsdokumentation

2004-02-27 Upprättad av Roland Thörner
INUSE 6.2/1.5.1
Prototyptestning 1, Per Mattsson, Henrik Ehrnlund
Prototyp in: ver 1 Prototyp ut: ver 1.1

Per och Henrik har nyligen avslutat sin magisterutbildning i Datorsystemteknik. De arbetar nu med att realisera ett avancerat (sekretessbelagt) RFID-system. Per och Henrik har mycket goda kunskaper inom programmering och protokoll kring RFID.

Jag presenterar applikationens övergripande idé, tänkt målgrupp etc.

Vi tittar på bild 1 och 2 i version 1 av RC version 1 och diskuterar frågan: Vad händer sedan?

Ur ett mer tekniskt perspektiv skall enligt Per och Henrik följande ske:

1. En OBDC-koppling måste göras, det är oklart om det kan åstadkommas i C eller C++, detta måste redas ut. Kan inte funktionen automatiseras måste användaren få bra stöd i detta, kanske i form av en guide. Kopplingen innefattar om den skall skötas manuellt en del abstrakt arbete.
2. I OBDC-kopplingen skall den databas man kopplar mot namnges, sker kopplingen automatiskt behöver användaren ändå ge ett namn. Ett defaultvärde här kan vara databasens filnamn exklusive filnamnstillägg.
3. I den databas man kopplar mot måste den kolumn väljas mot vilken RFID-identiteterna skall matchas. Här är det önskvärt att användaren får en lista med ingående kolumner/fält och där han/hon väljer ett alternativ.

Sedan har vi en diskussion om var viss annan funktionalitet bör läggas, i databasen eller RC. Per och Henriks uppfattniing är att den bör läggas i RC eftersom det ger möjlighet att stötta användaren på ett helt annat sätt. Alternativet är att användaren skriver kod i databasprogrammet, men det kommer att belasta användaren mycket mer. I installationsdelen av RC bör därför bland andra följande val göras:

1. Hur RC skall startas under vanlig drift:
    a. Startas manuellt
    b. Startas automatiskt när datorn startas
    c. Starta när RFID-läsaren identifierar en tagg

2. Vilket formulär som skall öppnas:
    a. Vid en känd RFID-identitet (finns i db)
    b. Vid en ny identitet

3. Vilka olika säkerhetsinställningar mm som skall göras, dvs vilka rättigheter som delas ut till användaren för RC.

I övrigt så diskuterar vi programmets övergripande struktur och är överens om att användaren bör uppleva två delar i programmet, en installationsdel där det förutsätts viss kunskap och en servicedel där användaren egentligen inte skall behöva vara medveten om att programmet körs.

Under sessionen blir det också uppenbart att nästa prototyp behöver skapas i en miljö som möjliggör funktionalitet. Inte minst påkallas detta av frågan som ofta dyker upp ”Vad händer sedan”.

RFID-Connect Systemutvecklingsdokumentation

2004-03-02 Upprättad av Roland Thörner
INUSE 6.2/1.5.1
Prototyptestning 2, Per Mattsson, Henrik Ehrnlund
Prototyp in: ver 2 Prototyp ut: ver 2.1

Vi går igenom den nya prototypen. Hela tiden med frågan: Är det här möjligt att göra? Två saker måste ovillkorligen rättas till:

1. Ordningsföljden i de olika stegen vid databaskopplingen . Här måste tyvärr det svåraste momentet, ODBC-kopplingen komma först eftersom det utgör en grund för att de kommande stegen 1-3 skall kunna automatiseras.

2. Under serviceinställningarna behöver alternativet, ”När RFID-läsaren aktiveras av tagg” arbetas om. Alternaivet innebär att ändå en lite programslinga behöver ligga och köra på datorn och det alternativet kommer sannolikt att bli väldigt likt alternativet ”Starta automatiskt”. Eftersom det blir frågan om två snarlika alternativ plockar jag bort det ena.

Under sessionen märker jag också att en del navigationsfrågor måste ses över. Förmodligen är det mest riktiga knappalternativet ”Fortsätt”, ”Tillbaka”, ”Avbryt” och ”Hjälp”.

Jag ser också att alternativen på startsidan behöver förtydligas. En något mer allvarlig invändning i första hand mot mig är att jag nog varit lite dålig på att förklara applikationens funktion. Sannolikt har jag överskattat killarnas kunskaper i just Access och vad man kan göra även i ett enkelt databasprogram som detta.

RFID-Connect Systemutvecklingsdokumentation

2004-03-02 Upprättad av Roland Thörner
INUSE 6.2/1.5.1
Prototyptestning 3, Tony Lundén
Prototyp in: ver 2.1 Prototyp ut: ver 2.2

Tony arbetar på CC-lab, undervisar i programmering och databaser. Frågan som är den mest intressanta just nu är om det går att automatisera en ODBC-koppling. Med Tony har jag en ganska omfattande genomgång.

Sessionen ger ett relativt tydligt besked och det är att det går att automatisera en ODBC-koppling, vilket innebär att prototypen får göras om på den här punkten. Vi har dessutom en ganska omfattande diskussion kring åtkomsträttigheter och vad man kan nå om databasen ligger i ett nätverk utan att behöva logga in på värddator eller databas alltför ofta.

De här frågorna måste redas ut och det kan vara ett lämpligt steg i nästa session.

Bilaga 3, Intervjuguide

Inledande (30 min)

Efter inledningen skall det psykologiska fältet vara öppnat, vi skall känna respondenten och han oss. Inga oklarheter skall råda kring situationen, tidsmässiga och andra ramar skall vara utredda. Atmosfären skall vara lugn och avslappnad. Respondenten skall ges ordentligt utrymme att prata. Om nödvändigt tar vi några ytterligare frågor kring arbete, teknik, vanliga tillämpningar, kunder et cetera.

1. Vi presenterar oss, personligt, studiemässigt, professionellt.
2. Praktiska saker:
    a. Får vi spela in intervjun på band?
    b. Intervjun kommer att ta ca 2 timmar, är det OK om vi drar över något
    c. Något annat som påverkar oss under intervjun?
    d. Hur vill du att vi behandlar ditt namn och företagsuppgifter i uppsatsen?
    e. Vi kommer att besöka flera företag, kort sekretesspolicy
3. Vi beskriver:
    a. Vårt arbete på ett övergripande plan
    b. Idén med programvaran beskrivs översiktligt
    c. Vad resultatet av arbetet skall bli.
4. Kan du berätta för oss vad du arbetar med? Din bakgrund?
5. Har du några frågor till oss som gäller hanteringen av uppgifter eller något annat som känns oklart kring vad vi vill göra?

Närmare presentation och frågor kring programvaran (20 min)

Efter presentationen skall respondenten vara helt på det klara med vad vi vill uppnå och vad vi har för tankar kring tänkta användningsområden. Han skall också vara medveten om sin egen roll i undersökningen. Vi är noga med att hela tiden inte ge respondenten för mycket information om hur vi själva hittills tänkt.

1. Vi beskriver den tänkta programvaran
    a. med ord
    b. med vår presentationsbild
2. När vi beskriver det vi vill göra så här;
    a. förstår du då hur vår tänkta mellanliggande programvara kan se ut och vilka tjänster den kan erbjuda?
    b. vilka frågor och oklarheter tycker du då kommer upp?
3. Skulle du kunna ha/ha haft nytta av en sådan programvara. Kan du i så fall beskriva de situationer där den skulle vara en tillgång?
4. Vem ser du som användare av en sådan programvara?
5. Vi bedömer att du är en viktig person för oss i vårt arbete
    a. Hur ser du på dig själv som lämplig intervjuperson i det här arbetet?
    b. Vilka av dina styrkor och svagheter inom området tycker du vi skall vara medvetna om i vårt fortsatta arbete?
6. Sammanfatta respondentens syn på sin egen roll

Fördjupad intervju (40 min)

I den fördjupade delen av intervjun kommer vår fingertoppskänsla till pass. Vi måste här få respondenten att delge oss sina erfarenheter. De tematiserade frågorna skall inte gås igenom en och en utan snarare väljer vi en eller ett par inom varje område som vi diskuterar djupt kring. Papper och penna finns framme. Vi visar de första delarna av prototypen. Respondenten måste uppleva att han är experten, vi håller en "lagom" profil, vilket innebär att vi inte pratar på för mycket och bara bekräftar vår egen uppfattning. Få gärna respondenten att tänka högt. Låt respondenten förstå att han är en mycket viktig person i arbetet.

Fördjupade frågor inom Quality in use

a. Functionality
b. Reliability
c. Usability
d. Efficiency
e. Maintainability
f. Portability

Tematiserade frågor:

Fördjupade frågor inom funktionalitet

- Om du med dina erfarenheter skulle skapa en sådan här programvara, vilken funktionalitet skulle du då lägga in i den och varför?
- Om vi tänker oss det scenario du beskrev förut, där du bedömde att du skulle ha haft nytta av den här mellanliggande programvaran. Hur skulle den se ut och vilka tjänster skulle den erbjuda för att nyttan skulle bli så stor som möjligt?
- Ser du några specifika användningsområden för en programvara som denna och vilken funktionalitet skulle i så fall en sådan behöva ha?

Fördjupade frågor inom användarvänlighet/funktionalitet

- Vad betyder begreppen funktionalitet och användarvänlighet för dig och vilka samband ser du mellan begreppen i det här sammanhanget?
- Om det finns ett motsatsförhållande mellan funktionalitet och användarvänlighet, hur skulle du då i så fall prioritera i det här fallet?

1. Sammanfatta respondentens svar.
2. Vad har vi missat eller inte förstått riktigt tycker du?
3. Hur känns det nu när du ser hela resultatet av din tankemöda?
4. Skulle en sådan här programvara bidra till att fler skapar informationssystem grundade på RFID?
5. Skulle du vara intresserad av att delta i utveckling av en sådan här programvara?

Avslutande frågor (20 min)

Nu är det viktigt att återgå till vanligt samtal igen. Respondenten måste känna att vi tar allt på allvar och att vi förvaltar hans kunskaper väl. Han får inte heller känna att han lämnat ifrån sig något han ångrat.

1. Nu har vi fått mycket att fortsätta med i vårt arbete, vad tycker du om relevansen i vårt arbete?

2. Har samtalet väckt några andra funderingar hos dig, borde vi fokusera helt eller delvis på någon annan fråga?
3. Är det något som framkommit under samtalet som du vill kommentera särskilt? Har du någon känsla av att vi missförstår varandra på någon punkt eller är det något av det du sagt som inte skall föras vidare?
4. Vi skickar naturligtvis en kopia på vårt arbete när det är klart.

Bilaga 4, Intervjuformulär

Fördjupade frågor inom external quality

Functionality - The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för functionality?

Reliability - The capability of the software to maintain its level of perform ance when used under specified conditions.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för reliability?

Usability - The capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för usability?

Efficiency - The capability of the software to provide the required perform ance, relative to the amount of resources used, under stated conditions.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för efficiency?

Maintainability - The capability of the software to be modified. Modification may include corrections, improvements or adaptation of the soft ware to changes in environment, and in requirements and functional specification.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för maintainability?

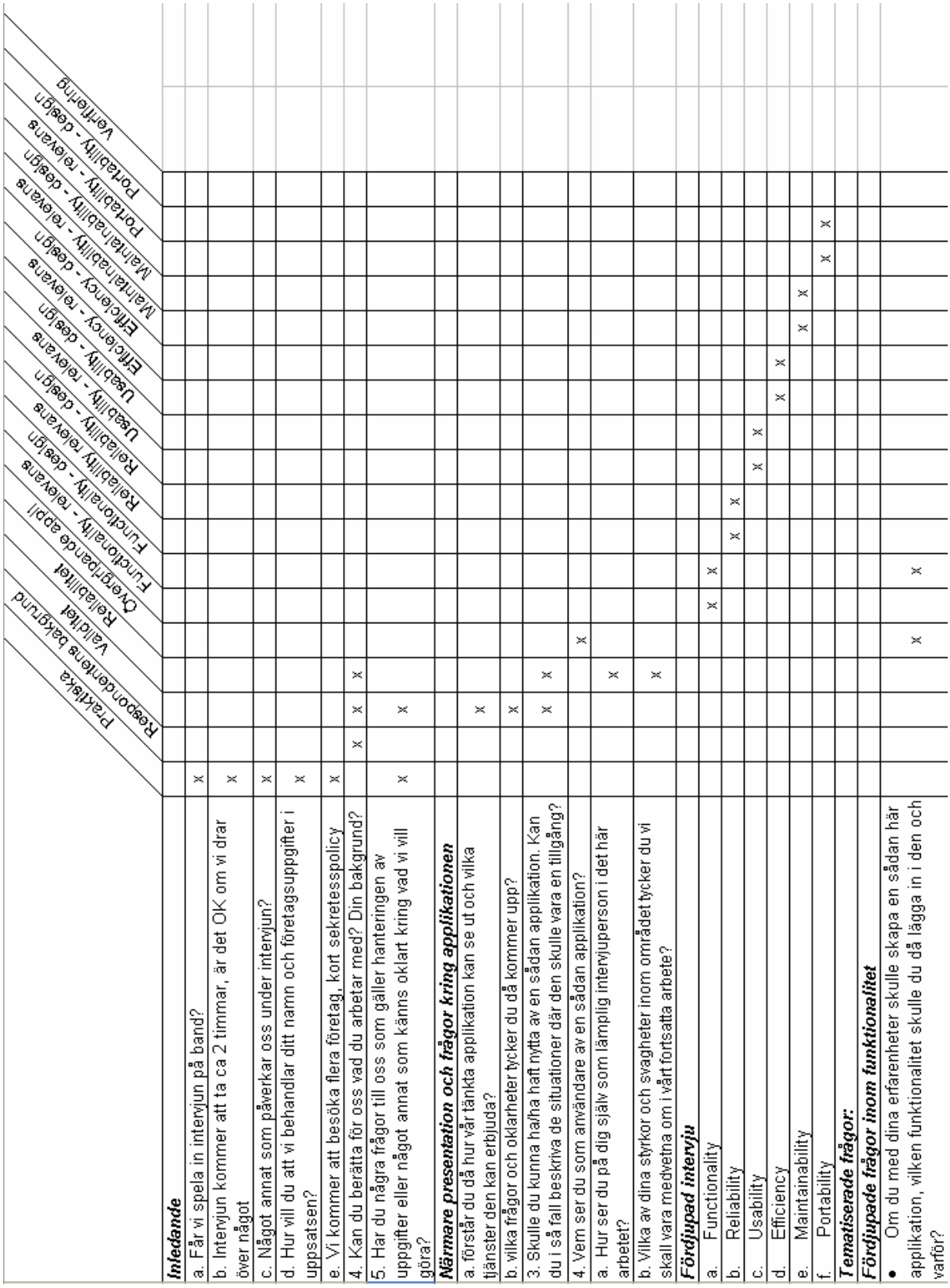
Portability - The capability of software to be transferred from one environ ment to another.

1. Hur bedömer du detta begrepps relevans för en programvara som denna:

Saknar relevans Liten relevans Stor relevans Mycket stor relevans

2. Kan du ge exempel på vad vi bör ta hänsyn till för att designa för portability?

Bilaga 5, Analysmall



	Praktiska	Respondenternas bakgrund	Validitet	Reliabilitet	Övergripande applikation	Functionality - relevans	Functionality - design	Reliability - relevans	Reliability - design	Usability - relevans	Usability - design	Efficiency - relevans	Efficiency - design	Maintainability - relevans	Maintainability - design	Portability - relevans	Portability - design	Verifiering
Inledande																		
a. Får vi spela in intervjun på band?	x																	
b. Intervjun kommer att ta ca 2 timmar, är det OK om vi drar över något	x																	
c. Något annat som påverkar oss under intervjun?	x																	
d. Hur vill du att vi behandlar ditt namn och företagsuppgifter i uppsatsen?	x																	
e. Vi kommer att besöka flera företag, kort sekretesspolicy	x																	
4. Kan du berätta för oss vad du arbetar med? Din bakgrund?		x	x	x														
5. Har du några frågor till oss som gäller hanteringen av uppgifter eller något annat som känns oklart kring vad vi vill göra?	x		x															
Närmare presentation och frågor kring applikationen																		
a. förstår du då hur vår tänkta applikation kan se ut och vilka tjänster den kan erbjuda?		x																
b. vilka frågor och oklarheter tycker du då kommer upp?		x																
3. Skulle du kunna hafha haft nytta av en sådan applikation. Kan du i så fall beskriva de situationer där den skulle vara en tillgång?		x	x															
4. Vem ser du som användare av en sådan applikation?				x														
a. Hur ser du på dig själv som lämplig intervjuperson i det här arbetet?			x															
b. Vilka av dina styrkor och svagheter inom området tycker du vi skall vara medvetna om i vårt fortsatta arbete?			x															
Fördjupad intervju																		
a. Functionality					x	x												
b. Reliability								x	x									
c. Usability										x	x							
d. Efficiency												x	x					
e. Maintainability														x	x			
f. Portability																x	x	
Tematiserade frågor:																		
Fördjupade frågor inom funktionalitet																		
• Om du med dina erfarenheter skulle skapa en sådan här applikation, vilken funktionalitet skulle du då lägga in i den och varför?					x		x											

	Praktiska	Bakgrund	Validitet	Reliabilitet	Övergripande	Funktionalitet	Funktionalitet - relevans	Reliabilitet - design	Reliabilitet relevans	Usability - design	Usability - relevans	Efficiency - design	Efficiency - relevans	Maintainability - design	Maintainability - relevans	Portability - design	Portability - relevans	Verifiering
• Om vi tänker oss det scenario du beskrev förut, där du bedömde att du skulle ha haft nytta av den här applikationen. Hur skulle den se ut och vilka tjänster skulle den erbjuda för att nyttan skulle bli så stor som möjligt?				x		x		x		x		x		x		x		
• Ser du några specifika användningsområden för en applikation som denna och vilken funktionalitet skulle i så fall en sådan här applikation behöva ha?				x		x												
Fördjupade frågor inom användarvänlighet/funktionalitet																		
• Vad betyder begreppen funktionalitet och användarvänlighet för dig och vilka samband ser du mellan begreppen i det här sammanhanget?						x				x								
• Om det finns ett motsatsförhållande mellan funktionalitet och användarvänlighet, hur skulle du då i så fall prioritera i det här fallet?					x	x			x	x								
2. Vad har vi missat eller inte förstått riktigt tycker du?		x	x															
3. Hur känns det nu när du ser hela resultatet av din tankemöda?				x														
4. Skulle en sådan här applikation bidra till att fler skapar informationssystem grundade på RFID?																	x	
5. Skulle du vara intresserad av att delta i utveckling av en sådan här applikation?		x	x														x	
Avslutande frågor																		
1. Nu har vi fått mycket att fortsätta med i vårt arbete, vad tycker du om relevansen i vårt arbete?		x	x	x														
2. Har samtalet väckt några andra funderingar hos dig, borde vi fokusera helt eller delvis på någon annan fråga?		x	x	x														
3. Är det något som framkommit under samtalet som du vill kommentera särskilt? Har du någon känsla av att vi missförstår varandra på någon punkt eller är det något av det du sagt som inte skall föras vidare?	x		x	x														

Bilaga 6, Referenstillämpningar

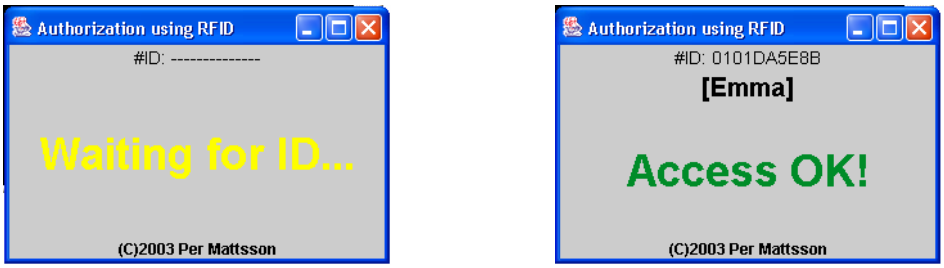
Som en del i arbetet har två referenstillämpningar grundad på en mellanliggande programvara skapats, en av vardera:

- Textfilskopplad
- Databaskopplad

Tillämpningarna har efter vår kravspecifikation programmerats av Per Mattson, magisterstudent i datorsystemteknik vid Högskolan i Halmstad.

Tillämpning 1 – Tillträdeskontroll, textfilskopplad

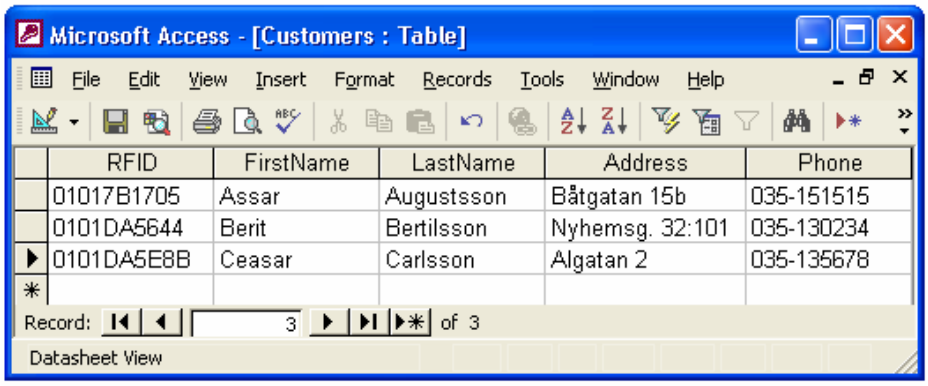
Den här tillämpningen kan användas för ett inpasseringssystem. Tillämpningen arbetar mot en textfil, dvs. de identiteter som skall ha tillträde finns lagrade i en fil som fungerar som systemets databas. Det här får sägas vara den enklaste typen av RFID-databas/tillämpning. I en riktig situation hade man sannolikt inte visat resultatet av en identifiering mot systemet på en bildskärm utan snarare styrt ett lås, en grind eller något motsvarande.



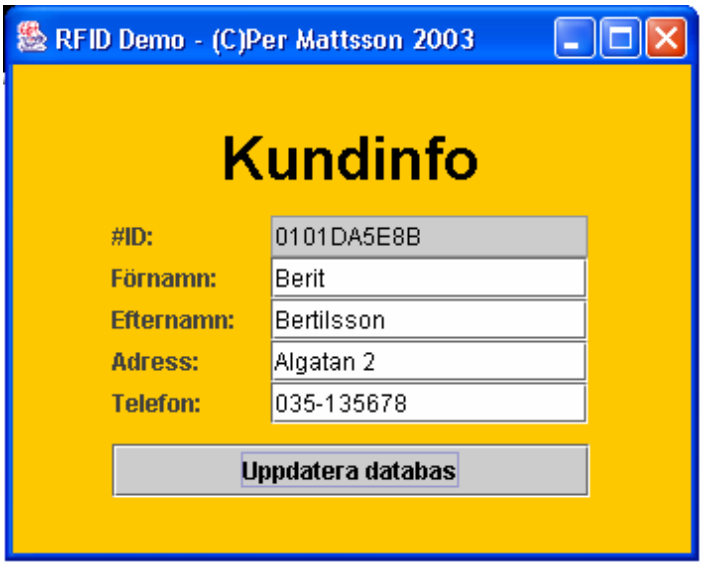
Figur 29. Gränssnitt från tillträdeskontrollsystemet. Emma äger genom sin RFID-identifiering tillträde. Återgivna med tillstånd ifrån Per Mattsson.

Tillämpning 2 - Medlemsregister, databaskopplad

Den databaskopplade tillämpningen har sin information lagrad i en databas. Just den här arbetar med ett formulär som skapas i Java. Primärnyckelfältet som här heter ”RFID” innehåller respektive RFID:s identitet.



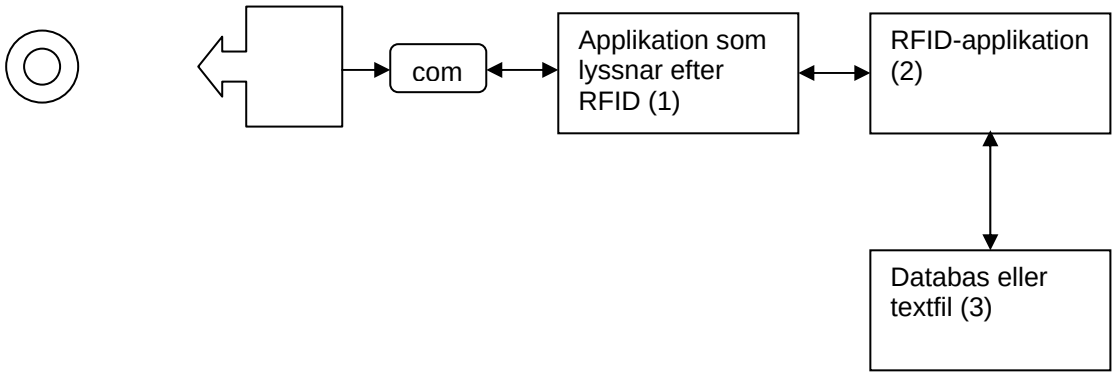
Figur 30. Innehållet i databasen som RFID-systemet är kopplat till. Återgivna med tillstånd ifrån Per Mattsson.



Figur 31. Javagränssnitt mot Accessdatabasen, kopplad via RFID. Återgivna med tillstånd ifrån Per Mattsson.

Programvara för koppling av RFID - databas

Johansson et al (2002) beskriver översiktligt processerna i ett system som hanterar RFID. Förenklat kan figur 32 åskådliggöra dessa skeenden. Efter samtal med Per Mattsson (2003) framgick följande punkter:



Figur 32. Schematisk bild över skeenden i RFID-konfigurerad dator.

1. Applikation 1 lyssnar av datorns seriella port (COM) efter signaler (identiteter) från RFID-läsaren.
2. Applikation 2 är den egentliga RFID applikationen som styr vad som skall ske när applikation 1 tagit emot en RFID-identitet och denna förts vidare
3. Applikation 2 måste kunna avgöra om det är en legitim identitet den tagit emot, uppgifterna stäms därför av mot antingen en textfil eller databas

Den mellanliggande programvaran i referenstillämpningarna har skapats i Java. Skälen till detta är enbart att det är ett språk som möjliggör en relativt snabb programutveckling (Skansholm, 2002). Att referenserna skapats i Java innebär inget generellt ställningstagande att språket skulle vara lämpligare än andra när det gäller att realisera en mellanliggande programvaran som den vi beskriver. Snarare tvärtom, det finns invändningar mot att skriva program som skall lyssna på datorns serieport i just Java. I bilaga 7 redovisas källkoden till de båda till-

lämpningarna. Tabell 7 visar vilka delar av källkoden som respektive tillämpning använder.

Tabell 7. Javafiler som används av respektive tillämpning.

Fil/modul	Används av:	
	Tillträdeskontroll	Medlemsregister
RFIDReader.java	X	X
RFIDListener.java	X	X
ErrorHandler.java	X	X
Database.java		X
BookingGUI.java		X
Booking.java		X
Authorization.java	X	
AuthorizationGUI.java	X	

Bilaga 7, Källkod Referenstillämpningar

RFIDReader.java

```
import java.io.*;
import java.util.*;
import javax.comm.*;

/**
 * Klass vars funktion är att lyssna på COM-porten efter data från
 * en RFID-läsare.
 *
 * @author Per Mattsson
 */
public class RFIDReader extends Thread implements SerialPortEventListener {
    static CommPortIdentifier portId;
    static Enumeration portList;

    private InputStream inputStream;
    private SerialPort serialPort;
    private Thread readThread;

    /* En vektor men objekt som är intresserade av att meddelas om ID:n
    kommer in på COM-porten */
    private Vector listeners;
    private ErrorHandler errorHandler;

    private final static int STX = 0x02;
    private final static int ETX = 0x03;
    private String idString = "";

   /**
    * Konstruktor som körs när klassen skapas
    * @param comPort Namnet på den COM-port där RFID läsaren är inkopplad (Ex: "COM1")
    * @param errorHandler Den klass som hanterar (fel-)meddelanden.
    */
    public RFIDReader(String comPort, ErrorHandler errorHandler) {
        this.errorHandler = errorHandler;
        init(comPort);
    }

    public RFIDReader(String comPort) {
        init(comPort);
    }

    /**
     * Letar upp och försöker initiera önskad COM-port.
     * @param comPort Den COM-som skall initieras
     */
    private void init(String comPort) {
        portList = CommPortIdentifier.getPortIdentifiers();

        while (portList.hasMoreElements()) {
            portId = (CommPortIdentifier) portList.nextElement();
            if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                if (portId.getName().equals( comPort )) {
                    try {
                        serialPort = (SerialPort) portId.open("RFIDReader", 2000);
                    } catch (PortInUseException e) {
                        errorHandler.showErrorMessageAndExit(comPort + " is in use.");
                        System.exit(-1);
                    }
                }
            }
        }
        if (serialPort == null) {
            errorHandler.showErrorMessageAndExit("Could not find: " + comPort);
            System.exit(-1);
        }
        try { inputStream = serialPort.getInputStream(); } catch (IOException e) {}
                try { serialPort.addEventListener(this); } catch (TooManyListenersEx-
ception e) {}
        serialPort.notifyOnDataAvailable(true);
        try {
            serialPort.setSerialPortParams(9600,
```

```
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);
        } catch (UnsupportedCommOperationException e) {}

        listeners = new Vector();
    }

    /**
     * När denna tråd körs ligger den bara och väntar på att data skall
     * komma in på COM-porten
     */
    public void run() {
        try { Thread.sleep(20000); } catch (InterruptedException e) {}
    }

    /**
     * Denna funktion körs varje gång det kommer in data på COM-porten
     * @param event Den typ av händelse som har kommmit på COM-porten
     */
    public void serialEvent(SerialPortEvent event) {
        switch(event.getEventType()) {
        case SerialPortEvent.BI:
            System.out.println("[BI]");
            break;
        case SerialPortEvent.OE:
            System.out.println("[OE]");
            break;
        case SerialPortEvent.FE:
            System.out.println("[FE]");
            break;
        case SerialPortEvent.PE:
            System.out.println("[PE]");
            break;
        case SerialPortEvent.CD:
            System.out.println("[CD]");
            break;
        case SerialPortEvent.CTS:
            System.out.println("[CTS]");
            break;
        case SerialPortEvent.DSR:
            System.out.println("[DSR]");
            break;
        case SerialPortEvent.RI:
            System.out.println("[RI]");
            break;
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
            System.out.println("[OUTPUT_BUFFER_EMPTY]");
            break;
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[20];

            try {
                int numBytes = -1;
                while (inputStream.available() > 0) {
                    numBytes = inputStream.read(readBuffer);
                }
                boolean startOfId = false, endOfId = false;
                int startByte = 0;
                int endByte = numBytes;
                for (int i=0; i<numBytes; i++) {
                    if (readBuffer[i] == STX) {
                        startOfId = true;
                        startByte = i+1;
                    } else if (readBuffer[i] == ETX) {
                        endOfId = true;
                        endByte = i-2;
                    }
                }
                String receivedBytes = new String(readBuffer, startByte, endByte-
startByte);
                checkIfIdStringIsComplete(receivedBytes, startOfId, endOfId);
            } catch (IOException e) {}
            break;
        }
    }
```

```
    /**
     * Registrerar intresse i denna RDID-läsarklass. Dvs, att om ett ID har
     * mottagits från COM-porten så skall det objekt som har registrerat intresse
     * meddelas.
     * @param listener Det objekt som är intresserat av av att meddelas om ID:n
     * kommer in på COM-porten.
     */
    public void registerInterest(RFIDListener listener) {
        listeners.add(listener);
    }

    /**
     * Kollar om de bytes som har mottagits på COM-porten bidrar till att ett helt
     * ID har mottagits. Om så är fallet skickas detta ID till de objekt som har
     * registrerat sig mha registerInterest(RFIDListener listener)
     * @param newBytes De nya bytes som har kommit in på COM-porten
     * @param isStartOfId <true> om de inkommnade byten är början av ett ID, annars
<false>
     * @param isEndOfId <true> om de inkommnade byten är de sista i ett ID, annars <fal-
se>
     */
    private void checkIfIdStringIsComplete(String newBytes, boolean isStartOfId, boolean
isEndOfId) {
        if (isStartOfId) {
            System.out.print("\n#ID = [" + newBytes);
            idString = newBytes;
        } else {
            System.out.print(newBytes + "]");
            idString = idString + newBytes;
        }
        if (isEndOfId)
            sendIdToListeners(idString);
    }

    /**
     * Skickar en inkommen ID-sträng till intresserade objekt
     * @param stringId ID-strängen
     */
    private void sendIdToListeners(String stringId) {
        for (int i=0; i<listeners.size(); i++) {
            ((RFIDListener)listeners.get(i)).userScanned(stringId);
        }
    }
}
```

RFIDListener.java

```
/**
 * Ett interface som måste implementeras av de klasser som vill
 * kunna registrera intresse (registerInterest) i klassen RFIDReader
 *
 * @author Per Mattsson
 */
public interface RFIDListener {

    /**
     * Metod som anropas när ett ID har lästs in i RFIDReader
     * @param id Inläst ID-sträng
     */
    public void userScanned(String id);
}
```

ErrorHandler.java

```
import javax.swing.JOptionPane;
import java.awt.Component;

/**
 * Klass som hanterar och visar (fel-)meddelanden
 *
 * @author Per Mattsson
 */
public class ErrorHandler {

    Component parentComponent;

    /**
```

```
     * Konstruktor.
     * @param parentComponent Den grafiska komponent som felmeddelanden skall
     * anpassas till.
     */
    public ErrorHandler(Component parentComponent) {
        this.parentComponent = parentComponent;
    }

    /**
     * Visar ett felmeddelande och avslutar sedan programmet
     * @param errorMessage Felmeddelandet
     */
    public void showErrorMessageAndExit(String errorMessage) {
        JOptionPane.showMessageDialog(parentComponent, errorMessage, "Fel", JOption-
Pane.ERROR_MESSAGE);
        System.out.println("Error: " + errorMessage);
        System.exit(-1);
    }

    /**
     * Visar ett felmeddelande
     * @param errorMessage Felmeddelandet
     */
    public void showErrorMessage(String errorMessage) {
        JOptionPane.showMessageDialog(parentComponent, errorMessage, "Fel",
JOptionPane.ERROR_MESSAGE);
        System.out.println("Error: " + errorMessage);
    }

    /**
     * Visar ett informationsmeddelande
     * @param infoMessage Informationsmeddelandet
     */
    public void showInfoMessage(String infoMessage) {
        JOptionPane.showMessageDialog(parentComponent, infoMessage, "Information", JOp-
tionPane.INFORMATION_MESSAGE);
    }
}
```

Database.java

```
import java.sql.*;
import java.util.Hashtable;

/**
 * Klass som hanterar hämtning och lagring från och till en databas. Databasen
 * måste ha en ODBC-koppling. Denna klass förutsätter att databasen innehåller
 * vissa fält.
 *
 * @author Per Mattsson
 */
public class Database {

    private Connection conn;
    private Statement stmt;
    private ErrorHandler errorHandler;

    public Database(String ODBCName, ErrorHandler errorHandler) {
        this.errorHandler = errorHandler;
        init(ODBCName);
    }

    public Database(String ODBCName) {
        init(ODBCName);
    }

    /**
     * Skapar en databaskoppling.
     * @param ODBCName
     */
    private void init(String ODBCName) {
        // Load JDBC driver
        try {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        } catch (Exception e) {
            errorHandler.showErrorMessageAndExit("Failed to load JDBC driver.");
            System.exit(-1);
```

```
        }

        // Get a connection
        try {
            conn = DriverManager.getConnection("jdbc:odbc:" + ODBCName);
            //System.out.println("Connection established!");
        } catch (Exception e) {
            errorHandler.showErrorMessageAndExit("Connection to database failed.");
            System.exit(-1);
        }
    }

    /**
     * Hämtar kundinformation från databasen
     * @param rfid Det ID som är kopplat till kunden
     * @return En tabell med kundinformation
     */
    public Hashtable getCustomerInfo(String rfid) {
        Hashtable customerInfo = new Hashtable();
        try {
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT Customers.FirstName, Custom-
ers.LastName, Customers.Address, Customers.Phone FROM Customers WHERE (Custom-
ers.RFID='"+ rfid +"')");
            while ( rs.next() ) {
              customerInfo.put("FirstName", rs.getString("FirstName"));
              customerInfo.put("LastName", rs.getString("LastName"));
              customerInfo.put("Address", rs.getString("Address"));
              customerInfo.put("Phone", rs.getString("Phone"));
            }
            // Close connection
            rs.close();
            stmt.close();
        } catch (Exception e) {
            e.printStackTrace();
            //errorHandler.showErrorMessageAndExit("Possible error in SQL-question.");
            return null;
        }
        return customerInfo;
    }

    /**
     * Sparar/uppdaterar kundinformation i databasen
     * @param customerInfo En tabell innehållande kundinformationen som skall lagras
     * @param update <true> om fältet skall uppdateras, <false> om ett nytt fält skall
skapas
     * @return <true> om sparningen/uppdateringen lyckades, annars <false>
     */
    public boolean storeCustomerInfo(Hashtable customerInfo, boolean update) {
        System.out.println("Update = " + update);
        System.out.println("INSERT INTO Customers VALUES('"+
(String)customerInfo.get("Rfid") +"','"
                                                        +
(String)customerInfo.get("FirstName") +"','"
                                                        +
(String)customerInfo.get("LastName") +"','"
                                                        +
(String)customerInfo.get("Address") +"','"
                                                        +
(String)customerInfo.get("Phone") +"')");
        try {
            stmt = conn.createStatement();
            if (update)
                stmt.executeUpdate("DELETE FROM Customers WHERE RFID='"+
(String)customerInfo.get("Rfid") +"';");
            stmt.executeUpdate("INSERT INTO Customers VALUES('"+
(String)customerInfo.get("Rfid") +"','"
                                                        +
(String)customerInfo.get("FirstName") +"','"
                                                        +
(String)customerInfo.get("LastName") +"','"
                                                        +
(String)customerInfo.get("Address") +"','"
                                                        +
(String)customerInfo.get("Phone") +"')");
        } catch (Exception e) {
            System.err.println("Possible error in SQL-question.");
```

```
            e.printStackTrace();
            return false;
        }
        return true;
    }
}
```

BookingGUI.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Klass som hanterar den grafiska delen i Boknings-programmet (Booking)
 *
 * @author Per Mattsson
 */
public class BookingGUI extends JFrame {
    JLabel lblID = new JLabel();
    JLabel lblFirstName = new JLabel();
    JLabel lblLastName = new JLabel();
    JLabel lblAddress = new JLabel();
    JLabel lblPhone = new JLabel();
    JTextField txtID = new JTextField();
    JTextField txtFirstName = new JTextField();
    JTextField txtLastName = new JTextField();
    JTextField txtAddress = new JTextField();
    JTextField txtPhone = new JTextField();
    JButton btnUpdateDb = new JButton();
    JLabel lblKunder = new JLabel();

    public BookingGUI(ActionListener al) {
        super("RFID Demo - (C)Per Mattsson 2003");
        this.setDefaultCloseOperation(3);
        lblID.setFont(new java.awt.Font("Dialog", 1, 12));
        lblID.setForeground(Color.darkGray);
        lblID.setMaximumSize(new Dimension(80, 17));
        lblID.setMinimumSize(new Dimension(80, 17));
        lblID.setPreferredSize(new Dimension(80, 17));
        lblID.setHorizontalTextPosition(SwingConstants.LEFT);
        lblID.setText("#ID:");
        this.getContentPane().setLayout(new GridBagLayout());
        lblFirstName.setFont(new java.awt.Font("Dialog", 1, 12));
        lblFirstName.setForeground(Color.darkGray);
        lblFirstName.setMaximumSize(new Dimension(80, 17));
        lblFirstName.setMinimumSize(new Dimension(80, 17));
        lblFirstName.setPreferredSize(new Dimension(80, 17));
        lblFirstName.setHorizontalTextPosition(SwingConstants.LEFT);
        lblFirstName.setText("Förnamn:");
        lblLastName.setFont(new java.awt.Font("Dialog", 1, 12));
        lblLastName.setForeground(Color.darkGray);
        lblLastName.setMaximumSize(new Dimension(80, 17));
        lblLastName.setMinimumSize(new Dimension(80, 17));
        lblLastName.setPreferredSize(new Dimension(80, 17));
        lblLastName.setHorizontalTextPosition(SwingConstants.LEFT);
        lblLastName.setText("Efternamn:");
        lblAddress.setFont(new java.awt.Font("Dialog", 1, 12));
        lblAddress.setForeground(Color.darkGray);
        lblAddress.setMaximumSize(new Dimension(80, 17));
        lblAddress.setMinimumSize(new Dimension(80, 17));
        lblAddress.setPreferredSize(new Dimension(80, 17));
        lblAddress.setHorizontalTextPosition(SwingConstants.LEFT);
        lblAddress.setText("Adress:");
        lblPhone.setFont(new java.awt.Font("Dialog", 1, 12));
        lblPhone.setForeground(Color.darkGray);
        lblPhone.setMaximumSize(new Dimension(80, 17));
        lblPhone.setMinimumSize(new Dimension(80, 17));
        lblPhone.setPreferredSize(new Dimension(80, 17));
        lblPhone.setHorizontalTextPosition(SwingConstants.LEFT);
        lblPhone.setText("Telefon:");
        txtID.setPreferredSize(new Dimension(160, 21));
        txtID.setEditable(false);
        txtFirstName.setMinimumSize(new Dimension(160, 21));
        txtFirstName.setPreferredSize(new Dimension(160, 21));
        txtLastName.setMinimumSize(new Dimension(160, 21));
        txtLastName.setPreferredSize(new Dimension(160, 21));
```

```
        txtAddress.setMinimumSize(new Dimension(160, 21));
        txtAddress.setPreferredSize(new Dimension(160, 21));
        txtPhone.setMinimumSize(new Dimension(160, 21));
        txtPhone.setPreferredSize(new Dimension(160, 21));
        btnUpdateDb.setPreferredSize(new Dimension(240, 27));
        btnUpdateDb.setText("Uppdatera databas");
        lblKunder.setFont(new java.awt.Font("Dialog", 1, 30));
        lblKunder.setText("Kundinfo");
        this.btnUpdateDb.addActionListener(al);
        this.getContentPane().setBackground(Color.orange);
        this.setForeground(Color.black);
        this.getContentPane().add(lblID,  new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(lblFirstName,   new GridBagConstraints(0, 2, 2, 1,
0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(lblLastName,   new GridBagConstraints(0, 3, 2, 1, 0.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(lblAddress,  new GridBagConstraints(0, 4, 2, 1, 0.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(lblPhone,  new GridBagConstraints(0, 5, 2, 1, 0.0, 0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(txtID,   new GridBagConstraints(2, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(txtFirstName,  new GridBagConstraints(2, 2, 1, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(txtLastName,   new GridBagConstraints(2, 3, 1, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(txtAddress,   new GridBagConstraints(2, 4, 1, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(txtPhone,   new GridBagConstraints(2, 5, 1, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 0, 0),
0, 0));
        this.getContentPane().add(lblKunder,  new GridBagConstraints(0, 0, 3, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(0, 0, 10,
0), 0, 0));
        this.getContentPane().add(btnUpdateDb,  new GridBagConstraints(0, 6, 3, 1, 0.0,
0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(10, 0, 0,
0), 1, 0));
    }

    public void setID(String id) { txtID.setText( id ); }
    public void setFirstName(String firstName) { setText(txtFirstName, firstName); }
    public void setLastName(String lastName) { setText(txtLastName, lastName); }
    public void setAddress(String address) { setText(txtAddress, address); }
    public void setPhone(String phone) { setText(txtPhone, phone); }

    public String getID() { return txtID.getText(); }
    public String getFirstName() { return getText(txtFirstName); }
    public String getLastName() { return getText(txtLastName); }
    public String getAddress() { return getText(txtAddress); }
    public String getPhone() { return getText(txtPhone); }

    private void setText(JTextField tf, String text) {
        if (text.equalsIgnoreCase(" "))
            tf.setText("");
        else
            tf.setText(text);
    }
```

```
    private String getText(JTextField tf) {
        if (tf.getText().equals(""))
            return " ";
        else
            return tf.getText();
    }
}
```

Booking.java

```
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Applikation som visar på möjligheterna att knyta data inläst från
 * en RFID-läsare till information i en databas
 *
 * @author Per Mattsson
 */
public class Booking implements RFIDListener, ActionListener {

    private Database db;
    private RFIDReader rfidReader;
    private BookingGUI gui;
    private ErrorHandler errorHandler;
    private boolean lastIdExistInDb = false;
    private String currentId;

    /**
     * Konstruktor. Initierar GUI och RFID-läsare.
     */
    public Booking() {
        gui = new BookingGUI(this);
        gui.setSize(350, 280);
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        gui.setLocation(screenSize.width/2-gui.getSize().width/2,
                                screenSize.height/2-gui.getSize().height/2);
        gui.setVisible(true);
        errorHandler = new ErrorHandler(gui);
        db = new Database("RfidDb", errorHandler);
        rfidReader = new RFIDReader("COM1", errorHandler);
        rfidReader.registerInterest(this);
        rfidReader.start();
    }

    /**
     * Denna funktion körs när ett ID har inkommit. Information knutet
     * till ID:t hämtas från en databas och presenteras i GUI:t
     * @param id Det ID:t som lästs in
     */
    public void userScanned(String id) {
        Hashtable customerInfo;
        currentId = id;
        gui.setID(id);
        String firstName, lastName, address, phone;
        customerInfo = db.getCustomerInfo(id);
        firstName = (String)customerInfo.get("FirstName");
        lastName = (String)customerInfo.get("LastName");
        address = (String)customerInfo.get("Address");
        phone = (String)customerInfo.get("Phone");
        if (firstName != null) {
            lastIdExistInDb = true;
            gui.setFirstName(firstName);
            gui.setLastName(lastName);
            gui.setAddress(address);
            gui.setPhone(phone);
            System.out.println("\nNamn: " + firstName + " " + lastName);
            System.out.println("Adress: " + address);
            System.out.println("Telefon: " + phone);
        } else {
            lastIdExistInDb = false;
            gui.setFirstName("");
            gui.setLastName("");
            gui.setAddress("");
```

```
            gui.setPhone("");
            System.out.println(" <-- Det finns ingen person i databasen som matchar det-
ta ID!");
        }
    }

    /**
     * Denn funktion körs när man klickar på knappen Update i GUI:t. Om det finns
     * information lagras denna i databasen.
     * @param e
     */
    public void actionPerformed(ActionEvent e) {
        if (currentId == null) {
            errorHandler.showInfoMessage("Ett RFID måste vara avläst för att kunna upp-
datera databasen!");
        } else {
            Hashtable customerInfo = new Hashtable();
            customerInfo.put("Rfid", gui.getID());
            customerInfo.put("FirstName", gui.getFirstName());
            customerInfo.put("LastName", gui.getLastName());
            customerInfo.put("Address", gui.getAddress());
            customerInfo.put("Phone", gui.getPhone());
            db.storeCustomerInfo(customerInfo, lastIdExistInDb);
        }
    }

    public static void main(String[] args) {
        new Booking();
    }
}
```

Authorization.java

```
import java.awt.*;
import java.io.*;
import java.util.*;

/**
 * Applikation som använder en RFID-läsare för att genomföra
 * identitetskontroll.
 *
 * @author Per Mattsson
 */
public class Authorization implements RFIDListener {
    static String idAllowed = "0101DA5E8B";

    private RFIDReader rfidReader;
    private AuthorizationGUI gui;
    private Timer timer;
    private int nbrOfTasks = 0;
    private Hashtable accessList;
    private ErrorHandler errorHandler;

    /**
     * Konstruktor. Initierar GUI och RFID-läsare.
     */
    public Authorization() {
        gui = new AuthorizationGUI();
        gui.setSize(300, 200);
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        gui.setLocation(screenSize.width/2-gui.getSize().width/2,
                        screenSize.height/2-gui.getSize().height/2);
        gui.displayStatus(gui.STATUS_WAITING);
        gui.setVisible(true);
        errorHandler = new ErrorHandler(gui);
        readAccessListFromFile();
        rfidReader = new RFIDReader("COM1", errorHandler);
        rfidReader.registerInterest(this);
        rfidReader.start();
    }

    /**
     * Denna funktion körs när ett ID har inkommit. ID:t jämförs med
     * de ID:n som initialt lästs in från fil. Om ID:t finns med där
     * så bevilljas tillträde, annars nekas tillträde. Resultat presenteras
     * i GUI:t
     * @param id Det ID:t som lästs in
```

```
     */
    public void userScanned(String id) {
        gui.displayId(id);
        if ( accessList.containsKey(id) ) {
            gui.displayStatus(gui.STATUS_OK);
            gui.displayName( (String)accessList.get(id) );
        } else {
            gui.displayStatus(gui.STATUS_ACCESS_DENIED);
            gui.displayName(null);
        }
        timer = new Timer();
        timer.schedule(new ResetScreenTask(), 3000);
        nbrOfTasks++;
    }

    /**
     * Klass vars ända uppgift är att rensa skärmen önskad tid efter att
     * tillträdesinformation har visats.
     */
    private class ResetScreenTask extends TimerTask {
        public void run() {
            nbrOfTasks--;
            if (nbrOfTasks == 0) {
                gui.displayId(null);
                gui.displayName(null);
                gui.displayStatus(gui.STATUS_WAITING);
                timer.cancel();
            }
        }
    }

    /**
     * Läser in information om vilka ID:n som skall beviljas tillträde
     */
    private void readAccessListFromFile() {
        try {
            BufferedReader in = new BufferedReader(new FileReader("access.dat"));
            StringTokenizer st;
            accessList = new Hashtable();
            String line, idString, name;
            System.out.println("The following ID's are permitted access:");
            while ((line = in.readLine()) != null) {
                if (!line.startsWith("*")) {
                    st = new StringTokenizer(line,"-,");
                    if (st.hasMoreTokens()) {
                        idString = st.nextToken();
                        if (st.hasMoreTokens()) {
                            name = st.nextToken();
                        } else
                            name = "";
                        accessList.put(idString, name);
                        System.out.println("[" + idString + "]");
                    }
                }
            }
            in.close();
        } catch (IOException e) {
            System.out.println("Couldn't find the file: access.dat");
            System.exit(-1);
        }
    }

    public static void main(String[] args) {
        new Authorization();
    }
}
```

AuthorizationGUI.java

```
import java.awt.*;
import javax.swing.*;

/**
 * Klass som hanterar den grafiska delen i ID-kontrollprogrammet (Authorization)
 *
 * @author Per Mattsson
 */
```

```
public class AuthorizationGUI extends JFrame {
    public static final int STATUS_ACCESS_DENIED = -1;
    public static final int STATUS_WAITING = 0;
    public static final int STATUS_OK = 1;

    JLabel lblStatus = new JLabel();
    JLabel lblCopyright = new JLabel();
    JPanel northPanel = new JPanel();
    JLabel lblId = new JLabel();
    JLabel lblName = new JLabel();

    public AuthorizationGUI() {
        super("Authorization using RFID");
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    private void jbInit() throws Exception {
        this.setDefaultCloseOperation(3);
        this.getContentPane().setLayout(new BorderLayout());
        lblStatus.setFont(new java.awt.Font("Dialog", Font.BOLD, 34));
        lblStatus.setForeground(Color.yellow);
        lblStatus.setHorizontalAlignment(SwingConstants.CENTER);
        lblStatus.setHorizontalTextPosition(SwingConstants.CENTER);
        lblStatus.setText("Status");
        lblCopyright.setHorizontalAlignment(SwingConstants.CENTER);
        lblCopyright.setText("(C)2003 Per Mattsson");
        lblId.setFont(new java.awt.Font("Dialog", 0, 14));
        lblId.setHorizontalAlignment(SwingConstants.CENTER);
        lblId.setText("#ID: --------------");
        northPanel.setLayout(new BorderLayout());
        lblName.setFont(new java.awt.Font("Dialog", 1, 20));
        lblName.setHorizontalAlignment(SwingConstants.CENTER);
        lblName.setText("");
        this.getContentPane().add(lblStatus, BorderLayout.CENTER);
        this.getContentPane().add(lblCopyright, BorderLayout.SOUTH);
        this.getContentPane().add(northPanel, BorderLayout.NORTH);
        northPanel.add(lblId, BorderLayout.CENTER);
        northPanel.add(lblName, BorderLayout.SOUTH);
    }

    public void displayId(String id) {
        if (id == null)
            lblId.setText("#ID: --------------");
        else
            lblId.setText("#ID: " + id);
    }

    public void displayName(String name) {
        if (name == null || name.equalsIgnoreCase(""))
            lblName.setText("");
        else
            lblName.setText("[" + name + "]");
    }

    public void displayStatus(int status) {
        switch (status) {
            case STATUS_ACCESS_DENIED:
                lblStatus.setForeground(Color.red);
                lblStatus.setText("Access denied!");
                break;
            case STATUS_WAITING:
                lblStatus.setForeground(Color.yellow);
                lblStatus.setText("Waiting for ID...");
                break;
            case STATUS_OK:
                lblStatus.setForeground(new Color(0, 140, 40));
                lblStatus.setText("Access OK!");
                break;
        }
    }
}
```