KUNGL TEKNISKA HÖGSKOLAN
Royal Institute of Technology
Numerical Analysis and Computing Science

TRITA-NA-D0005 • CID-71, KTH, Stockholm, Sweden 2000

Användarcentrerad systemutveckling, version 1.0

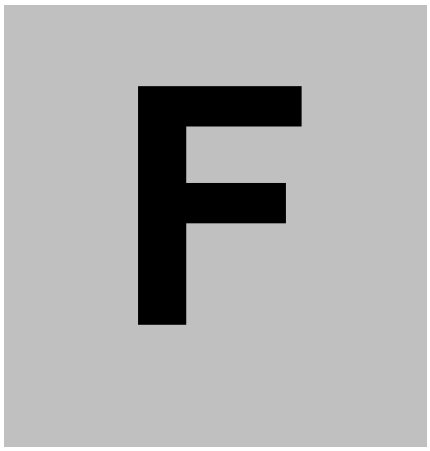
Bengt Göransson och Jan Gulliksen

med benägna bidrag från:
Jan Andersson, RSV IT
Inger Dahlgren, TietoEnator AB
Kjellåke Henriksson, RSV SK/BU
Ann Lantz, CID/KTH
Bengt Sandblad, avd. för MDI, Uppsala universitet
Yngve Sundblad, CID/KTH

CID
Centre for
User Oriented IT Design

Bengt Göransson och Jan Gulliksen
Användarcentrerad systemutveckling, version 1.0
Report number: TRITA-NA-D0005, CID-71
ISSN number: ISSN 1403-073X
Publication date: January 2000
E-mail of author: Jan.Gulliksen@hci.uu.se
URL of author: http://cid.nada.kth.se/cid/

Reports can be ordered from:
CID, Centre for User Oriented IT Design
Nada, Dept. Computing Science
KTH, Royal Institute of Technology
S-100 44 Stockhom, Sweden
telephone: + 46 8 790 91 00
fax: + 46 8 790 90 99
e-mail: cid@nada.kth.se
URL: http://www.nada.kth.se/cid/



Förord

Avdelningen för Människa-datorinteraktion (MDI) vid Uppsala universitet (tidigare Centrum för studium av människan och datorn – CMD) är en oberoende forskningsavdelning som sedan 1994 har bedrivit samverkansprojekt med Riksskatteverket – RSV i syfte att bidra till att öka användbarheten i de datorsystem som organisationen utvecklar samt att förbättra de arbetsprocesser som utvecklingen följer.

Under åren har vi deltagit i projekt som KONTUR, INDEX, FILBYTER, KONTIKI, MÅLBILDA, AKTIV, MAGI, SKATTEKONTO, INIT etc. Vi har varit drivande bakom framtagandet av RSVs Style Guide för gränssnittsutformning, där bland annat rumsmetaforen (arbetsytor) är en viktig del.

Sedan hösten 1998 har projektet »Användarcentrerad systemutveckling vid RSV« bedrivits med syftet att utreda och förbättra RSVs användarcentrerade systemutvecklingsmodell. Detta arbete bedrevs genom ett antal möten och modelleringar kring årsskiftet 1998/1999 kompletterat med intervjuer och utredningar under våren 1999. Huvudsakligen deltog från RSV Kjellåke Henriksson, Kristina Bergh och Jan Andersson, från MDI deltog Jan Gulliksen och Bengt Göransson. Från RSVs sida dokumenterades arbetet i rapporten

»*Använda användare i utveckling*« av Kjellåke Henriksson (RSV SK/BU) m. fl. och från vår sida sett kommer nu denna rapport.

Denna rapport beskriver grunderna i användarcentrerad systemutveckling och hur detta förhåller sig till systemutvecklingsmodellerna. Den beskriver användarmedverkan, projektstyrningsaspekter, metoder och roller i arbetet. Den relaterar hela tiden till det arbete som bedrivs på RSV och rapporterar också om ett antal observationer som gjorts under de fem år av samverkan med RSV som vi haft. Särskild tonvikt har lagts på att användarmedverkan skall vara effektiv i systemutvecklingsprocessen.

Syftet med denna rapport är att förklara varför ett användarcentrerat synsätt är nödvändigt i organisationen, vad man vinner på detta och vilka fallgropar man skall undvika. Det kan tjäna som ett underlag för att bedriva ett användarcentrerat utvecklingsprojekt, men också som underlag/krav för upphandling av en systemutvecklingsmodell och vad en sådan modell bör kompletteras med för att bli användarcentrerad. Den riktar sig till verksamhetsföreträdare och IT-strateger på låg som hög nivå.

Vi framför vårt tack till de organisationer och finansiärer som gjort detta arbete möjligt: RSV (Riksskatteverket), RALF (Rådet för arbetslivsforskning, inom ramen för projekt med diarienummer 1998-0307) och CID (Centrum för användarorienterad IT-design).

Vi vill också passa på och tacka följande personer för de texter som vi valt att inkludera för att beskriva vissa aspekter rörande användarcentrering; Inger Dahlgren, TietoEnator AB; Ann Lantz, CID NADA/KTH, Stockholm; Bengt Sandblad, avdelningen för MDI, Uppsala universitet; Yngve Sundblad, CID/KTH, Stockholm och Jan Andersson, RSV IT.

LÄSANVISNINGAR. Rapporten är inte tänkt att läsas från »pärm till pärm«. Vi tror att man får ut mesta möjliga av rapporten genom att välja ut de kapitel man tycker låter mest intressanta. Även om de exempel och studier som finns redovisade i rapporten i huvudsak bygger på arbete genomfört inom ramen för verksamhet inom RSV, tror vi att läsarna kan ha ett generellt utbyte av rapporten i termer av kunskap och erfarenheter kring användarcentrerad systemutveckling.

Jan Gulliksen och Bengt Göransson

Uppsala januari 2000

INNEHÅLLSFÖRTECKNING

1

Bakgrund till området

Det torde stå klart för de flesta att användningen av datorstöd i arbetslivet idag inte fungerar så bra som den tekniska utvecklingen borde ha lett till. Datorer används allt mer i arbetslivet och i samhället i övrigt. 1977 användes ca 300 000 datorer i arbetslivet, 1995 användes 2 100 000 datorer. 1998 kom mer än 73% i daglig kontakt med datorer i sitt arbete och 51% hade tillgång till internet hemma och/eller på jobbet. 34% sade sig hämta information på »nätet« minst en gång per vecka [Teledok, 1999].

1.1 Kostnader för dåliga datorstöd

I ett företag med 13 000 datoranvändare kostar det 250 000 kr per dygn i »ineffektivitetskostnader« räknat på genomsnittliga lönekostnader om man förlorar 10 minuter per användare på grund av dålig effektivitet i datorstödet. Detta skall givetvis inte ses som ett argument mot datoriseringen. Användarna av datorstöd i sitt arbete är förmodligen mycket effektivare än vad de skulle vara utan datorer och de utför förmodligen en stor mängd arbetsuppgifter som vore omöjliga utan datorstöd. Snarare än att räkna på de förlorade lönekostnaderna måste man förstås se till de konsekvenser som den dåliga datoriserade arbetsmiljön innebär. Sjukskrivningarna ökar, ofta med diagnosen

utbrändhet. Känslan av otillräcklighet, kompetensbrist, fysiska belastningsbesvär, etc ökar lavinartat.

Det är med andra ord mycket väl värt tiden att satsa på att de datorstöd som tas fram är effektiva, stabila, minimerar fel, kort sagt att vi har användbara datorstöd. En yrkesarbetare vill göra ett bra jobb effektivt och med kvalitet, inte lära sig att bli en datoroperatör. Datorstödet är bara bra så länge som det låter användarna »göra jobbet«.

CAP Gemini har gjort en undersökning på 975 personer som visar att en genomsnittlig PC-användare tillbringar 2 timmar och 22 minuter per vecka på att fundera över hur olika datorproblem skall kunna lösas. Summerat ger detta två och en halv veckas förlorad tid per år och arbetare till följd av strul och funderingar kring datorprogram (Veckans Affärer, 23/3 1998). Gartner Group i USA har utvecklat en metod för att räkna på synliga och osynliga datorkostnader och konstaterar att för ett storföretag som använder Windows 95 blir kostnaden för varje enskild arbetsplats 10 000 US$ dvs ca 85 000 kr per år, varav ca en tredjedel är kapitalkostnader, resten är kostnader för support och strul.

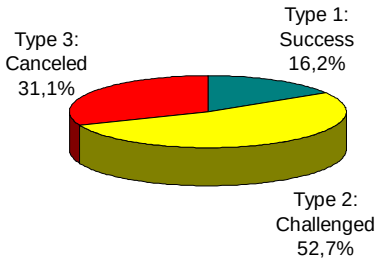
1.2 Problem i IT-utvecklingsprojekt

Givetvis kommer fokus snart att hamna på den process i vilket datorstödet tas fram. IT-utvecklingsprojekt är kända för att ofta försenas, fördyras och t o m avbrytas utan uppnått resultat. Enligt en nyligen utförd studie av London School of Economics är företagsledare besvikna på de investeringar man gjort på IT, och särskilt missnöjda är man i Norden. I en färsk intervjuundersökning har man intervjuat 500 verkställande direktörer och IT-chefer i 500 av de ledande företagen i världen. De uppger att bara 28 % av de pengar som satsas på IT förbättrar företagets lönsamhet. Nästan 60 % av IT-satsningarna syftar till att skapa konkurrensfördelar, men bara 37 % av investeringarna lever upp till den förhoppningen.

Är det då så enkelt att bara man blandar in användarna i processen så får man användbara system och följaktligen en bra arbetsmiljö? Givetvis inte, men man kan lära något av de undersökningar som gjorts om framgången hos IT-utvecklingsprojekt. Standish Group har i sin CHAOS-rapport [Standish Group, 1995] gjort en undersökning av amerikanska IT-utvecklingsprojekt. I USA spenderas 250 miljarder dollar varje år på 175 000 olika IT-projekt.

365 IT-företag med sammanlagt 8 380 olika IT-projekt undersöktes och följande kunde konstateras:

- 16,2 % genomfördes planenligt.
- 52,7 % genomfördes med förändrade planer.
- 31,1 % av företagens IT-projekt avbröts.



Figur 1. CHAOS report.

I snitt ökade kostnaderna för de ändrade planerna med 189 %. 81 miljarder dollar spenderas följaktligen varje år på projekt som aldrig blir färdiga.

Det finns ingen anledning att tro att siffrorna skulle ha blivit särskilt mycket bättre sedan 1995 eller att det skulle vara särskilt mycket annorlunda i Sverige, tvärtom uttrycker de organisationer som jag hittills visat dessa siffror för att det mycket väl stämmer överens med de IT-utvecklingsprojekt som utförts inom deras egen organisation. Den amerikanska undersökningen gick djupare i att försöka finna vari framgångsfaktorerna fanns för de 16,2 % »lyckade« projekten, och en av de aspekter som kom allra högst på listan var effektiv användarmedverkan i utvecklingsarbetet, en tydlig kravspecifikation var en annan. Bra planering och stöd från beslutsfattarna tillhörde också det viktigaste aspekterna.

Vad blir då de praktiska konsekvenserna av de dåliga oddsen för att lyckas med sina IT-utvecklingsprojekt? I media rapporteras om att chefer är missnöjda med IT-utvecklingen. Vad är den naturliga reaktionen hos dessa chefer? Jo man vågar inte riktigt att satsa på de åtgärder som är nödvändiga för att nå framgångsrika resultat.

1.3 Organiserad eller oorganiserad användbarhet i systemutvecklingsarbete

Närmare 50% av programkoden i ett IT-system utgörs av användargränssnittet (enligt t ex Nielsen, 1993). Motsvaras denna kod av 50% av utvecklingsarbetet? Mycket sällan ägnas gränssnittet så mycket tid. Oftast är det verksamhetsfunktionerna och databasarbetet som är det som tilldelas allra mest resurser i form av bemanning och utvecklingsresurser.

Hur stor andel av utvecklingsbudgeten ägnas då åt användbarhetsrelaterat arbete? Ett snabbt överslag hos flera stora svenska organisationer ger vid handen att andelen användbarhetsrelaterat arbete ofta inte uppgår till mer än en procent.

Hur används då denna procent? Trots att det nu börjar finnas viss kompetens inom detta område på arbetsmarknaden så är det få organisationer som bygger upp sin egen kompetens inom detta område. Om man väljer att bygga upp egen kompetens inom detta område så finns det inte direkt något stöd för hur denna kompetens skall se ut eller hur arbetet skall organiseras. Att skapa en användbarhetsorganisation inom ett existerande systemutvecklande företag är en forskningsfrågeställning som vore värdefull att ägna ytterligare studier.

I arbetet att bygga upp en organisation för användbarhetsrelaterat arbete är det viktigt att beakta dess position och befogenhet i organisationen. För det första så måste organisationen slå fast i sina övergripande mål att det är viktigt att bedriva arbetet på ett användarcentrerat sätt. För det andra är det viktigt att användbarhetskompetensen också finns med i organisationens ledning eftersom det ofta är i de strategiska besluten på hög nivå som mycket av ramarna för detta arbete sätts. Detta gäller inte bara i IT-utvecklingen utan i verksamhetsutvecklingen som sådan, kompetensutveckling, organisationsutveckling, etc. För att nå framgångsrik utveckling av IT skall man inte bara arbeta med användbarhetsförhöjande åtgärder i sluttampen av systemutvecklingsprocessen. Man måste få in ett fokus på att tillverka ***användbara*** system från början i allt utvecklingsarbete.

Den amerikanske forskaren Donald Norman hävdar att användbarhetskompetens i organisationer är fel använd, vilket innebär ett allvarligt misshushållande med resurserna. Idag är användbarhetskompetens ofta en resurs som används när processen så kräver, på initiativ av projektet. Till följd av detta kommer ofta utvärderingsinsatser för sent när det inte längre finns tid eller resurser till att göra nödvändiga förändringar. Användbarhetskompetensen borde vara ett kollegialt stöd i allt utvecklingsarbete (engelskans peers). Norman hävdar att användbarhetskompetensen själv är skyldig till detta genom att det finns för lite marknadstänkande och för lite kontakt med företagens marknadsavdelningar.

1.4 Användarmedverkan i systemutvecklingsprocessen

Sverige har unika möjligheter. Inget annat land i världen har kommit så långt vad gäller användarinflytande i systemutvecklingsprocessen, särskilt i tillverkandet av datorbaserat stöd för en arbetsuppgift. Användarens inflytande

över utvecklingen av sin arbetsmiljö och därigenom sitt IT-stöd är t o m säkerställd i arbetsmiljölagen (*se Figur 2*).

Arbetsmiljölagen (1977:1160) 2 kap. Arbetsmiljöns beskaffenhet, 1 §

- Arbetsmiljön skall vara tillfredsställande med hänsyn till arbetets natur och den sociala och tekniska utvecklingen i samhället
- Arbetsförhållandena skall anpassas till människors olika förutsättningar i fysiskt och psykiskt avseende.
- **Arbetstagaren skall ges möjlighet att medverka i utformningen av sin egen arbetssituation samt i förändrings- och utvecklingsarbete som rör hans eget arbete.**
- Teknik, arbetsorganisation och arbetsinnehåll skall utformas så att arbetstagaren inte utsätts för fysiska eller psykiska belastningar som kan medföra ohälsa eller olycksfall. Därvid skall även löneformer och förläggning av arbetstider beaktas. Starkt styrt eller bundet arbete skall undvikas eller begränsas.
- Det skall eftersträvas att arbetet ger möjlighet till variation, social kontakt och samarbete samt sammanhang mellan enskildas arbetsuppgifter.
- Det skall vidare eftersträvas att arbetsförhållandena ger möjlighet till personlig och yrkesmässig utveckling liksom till självbestämmande och yrkesmässigt ansvar.

Figur 2. Utdrag ur arbetsmiljölagen.

Varje arbetstagare har alltså en laglig rätt, och kanske rent av skyldighet att medverka i utformningen av sitt datorstöd. Hur ofta tillämpas eller åberopas denna lag? Skulle man rent av kunna stämma någon för att ens datorstöd har låg användbarhet och har tagits fram utan att medverkan från användarna skett?

Även om lagen inte praktiskt tillämpats så är den viktig för att påtala den viktiga roll användarna bör ha i utvecklingsarbetet.

Många svenska utvecklingsprojekt är tämligen välförsedda i termer av användarmedverkan i systemutvecklingen. Dock har vi ofta kunnat observera hur detta arbete skett ineffektivt, mållöst och utan att användarna egentligen getts möjlighet att medverka med det de är bra på. Därför har vi skapat denna

rapport för att dokumentera och påtala de brister vi sett i utvecklingsprocesserna så att framtida utveckling kan undvika dessa fallgropar. Rapporten sammanfattar nuläget forsknings- och utvecklingsmässigt samt anger hur användarcentrerad systemutveckling förhåller sig till olika kategorier av anställda inom organisationen och hur det förhåller sig till befintliga utvecklingsmodeller.

2

Användbarhet och användarcentrering

Att användarna skall stå i centrum vid utvecklingen av ett nytt datorstöd borde vara en självklarhet. Detta är dock tyvärr inte alltid fallet. Ofta utgår utvecklingen ifrån ett teknikcentrerat synsätt där tekniska lösningar får styra arbetssättet. Metoderna som används i systemutvecklingsarbetet är oftast av traditionell typ och baseras på exempelvis en vattenfallsmodell [Boehm, 1976]. Denna kategori av metoder stöder inte en utveckling av vad vi vill kalla användbara system, där användaren och dennes arbetsuppgifter ställs i centrum. En välfungerande arbetsmiljö kan lätt bli förstörd av en illa genomtänkt och införd datorisering.

Det finns ofta en övertro på att »bara man får en dator« så löser sig alla problem. Effekten blir ofta den motsatta och skapar frustration hos användarna samt nya och större problem i arbetssituationen. För att undvika en utveckling av denna typ anser vi att man bör välja en användarcentrerad utvecklingsmetodik.

Användbara och ändamålsenliga tillämpningar är inte något som bara uppstår. För att skapa dessa behövs metodik och en förståelse för användarna, deras

social miljö och arbetsuppgifter. Användarna skall involveras i alla steg av utvecklingsprocessen, inte bara »tycka till« om designförslag. Utgångspunkten för den användarcentrerade utvecklingen är användarnas behov, förutsättningar och möjligheter.

ANVÄNDBARHET är ett centralt begrepp i den användarcentrerade systemutvecklingsprocessen. *ISO 9241–11 Guidance on usability* definierar användbarhet som [ISO 9241–11, 1998]:

> »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.«

Produkten eller systemet skall alltså vara målrelaterat och:

- gå att använda *effektivt*,
- vara *produktivt* att använda,
- *accepterat* av användarna

– detta i en viss miljö eller visst sammanhang.

Orsaken till att vi väljer ISOs definition av användbarhet är att den tar ett mycket vidare grepp om användbarhet än tidigare definitioner. I takt med de allt ökande problemen med IT-stöd i arbetslivet som vi ser finner vi det värdefullt att ha en definition som försöker ta ett helhetsgrepp om problematiken. Dessutom, det faktum att det är en internationell standard gör det mycket enklare att oomtvistat införa ett sådant begrepp i en rigid organisation. Följande är de viktiga aspekterna i och med att man använder ISOs definition på användbarhet:

- Användbarheten enligt ISOs definition är ett betydligt vidare begrepp än så som användbarhet betraktas i dagligt tal. En av de viktiga utvidgningarna är att även systemets estetiska värden är av betydelse för användbarheten. I begreppet »context of use« avses också att inte bara vilka som använder systemet, vilken målsättning användaren har utan även det sammanhang i vilket systemet används är av betydelse för användbarheten.
- Ser på användbarhet som en *mätbar* storhet, dvs man skall kunna kvantifiera i vilken utsträckning något är användbart samt kunna ange användbarheten i relation till någon annan produkt. På så sätt kan man också avgöra om en insats har givit avsedd effekt i termer av förhöjd användbarhet. Det finns givetvis en stor mängd metoder för att kvantifiera de olika aspekterna av användbarhet och vissa av dessa kan man finna i

kapitlet om utvärderingsmetoder (se avsnitt, *3.7 Utvärderingsmetoder – med och utan användare*). Huvudsakligen är det effektiviteten som kan mätas i termer av tid för att utföra vissa arbetsuppgifter och ändamålsenligheten, bland annat genom att mäta felfrekvensen och tiden att återhämta sig från fel. Även användartillfredsställelsen kan mätas genom upprepade enkätstudier som påtalar användarupplevelserna i termer av tillfredsställelse och graden av hur estetiskt tilltalande systemet upplevs.

- Vidare är definitionen viktig i och med att den går att konkretisera och omsätta i metoder och inte minst gör det möjligt att fokusera på användbarhet i systemutvecklingsprocessen.

För att vara mer konkret kan man också använda ett antal punkter som användbarhet traditionellt förknippas med [Nielsen, 1993]:

- **Lätt att lära:** Så att användaren snabbt kommer igång med arbetet.
- **Effektivt att använda:** När användaren har lärt sig systemet måste det vara effektivt att arbeta med.
- **Lätt att komma ihåg:** Det måste gå att återkomma till systemet efter en tids frånvaro och ändå kunna komma ihåg hur det fungerar.
- **Få fel:** Användarna skall kunna göra så få fel som möjligt. Om man ändå gör fel måste det gå att komma tillbaka till situationen innan felet uppstod.
- **Tillfredsställande:** Det skall kännas »angenämt« att använda systemet. Man skall känna en tillfredsställelse att jobba med systemet, helt enkel tycka om det.

2.1 Kognition, beslutsfattande och modeller

En viktig del av utvecklingsarbete är att skapa en beskrivning, en modell, av »arbetssystemet«. Beskrivningen ska utgöra en ram för att förstå systemet, kunna resonera kring det samt kunna analysera det för att förstå viktiga aspekter på arbetet som bedrivs i det, informationshanteringen, krav på datorstöd m.m.

För att klara av detta behöver man först en grundläggande struktur för att kunna beskriva och analysera arbetssystemet. Systemet man skall beskriva är inte datorsystemet som sådant, utan hela arbetsmiljön i vilken arbete med datorstöd bara är en del.

För att förstå hur en hel arbetsmiljö för en användare kan te sig är det viktigt att förstå strukturen på det arbete som utförs. En stor del av det arbete som utförs på RSV kan beskrivas som ärendehanteringsarbete. Som beskrevs i AKTIV-

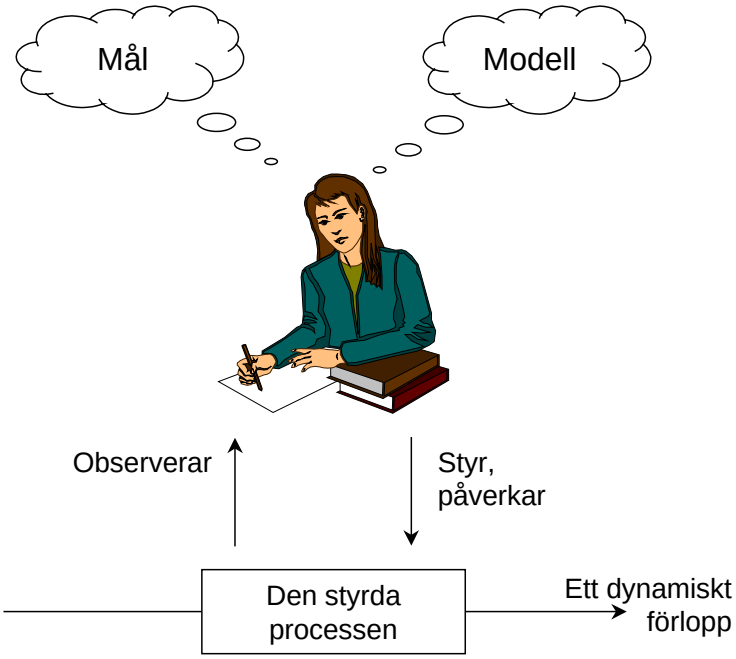
rapporten[1] är just det mänskliga beslutsfattandet en viktig del av allt ärendehanteringsarbete. Därför behöver vi kunna förstå hur mänskligt beslutsfattande går till och skapa en lämplig struktur för att beskriva, analysera och förstå viktiga aspekter på mänskligt beslutsfattande. Det visar sig att en fungerande sådan struktur kan hämtas från styr- och reglertekniken. Detta stämmer väl överens med system där även de tekniska och organisatoriska aspekterna på system och arbete ska kunna beskrivas och förstås.

Beskrivningsmodellens struktur. Ärendehantering kan ses som en process som förändrar och förädlar information om ett ärende utifrån ärendets speciella struktur. Flera olika faktorer måste vara uppfyllda för att man ska kunna förstå, övervaka och styra en sådan process. En grundläggande kategorisering av sådana faktorer har vi hämtat från vad styr- och reglertekniken säger om vad som fordras för att styra/kontrollera en process.

För styrning fordras att samtliga av följande villkor är uppfyllda:

- att det finns ett tydligt mål för det som ska uppnås,
- att den som ska styra/kontrollera har en modell över (förstår hur det fungerar, har kunskap om etc.) processen eller skeendet,
- att det finns tillräckliga möjligheter att påverka processen eller skeendet (det s.k. styrbarhetsvillkoret),
- att den som styr har tillräcklig information om processens eller skeendets aktuella tillstånd (det sk observerbarhetsvillkoret).

[1] AKTIV-rapporten är framtagen inom RSV och beskriver ett nytt sätt att strukturera ärendehanteringsprocessen. I stället för att betrakta varje tillämpning som ett »stuprör« som hanterar indata, handlägger ärenden, producerar utdata etc så försöker man etablera generella ärendehanteringsmoduler som »indatahantering«, »uppföljning« och »handläggning«. För dessa moduler kan sedan IT-stöd implementeras som lätt skall kunna hakas in i varandra. AKTIV-rapporten kan beställas från riksskatteverket eller arbetet kan läsas om vetenskapligt i Gulliksen (1996).



Figur 3. För att styra en process. fordras mål, modell, styrbarhet och observerbarhet. Dessutom är den process som ska styras oftast dynamisk, vilket innebär att systemets tillstånd förändras spontant och som följd av påverkan, samt att styråtgärder inte bara har effekt momentant utan även i framtiden.

I ett enkelt tekniskt sammanhang, t. ex. när det gäller att köra en bil, är det lätt att inse betydelsen av att villkoren enligt ovan är uppfyllda. Man måste ha målet för färden helt klart för sig, tillsammans med ytterligare mål som kan påverka förloppet, t. ex. som rör säkerhetsaspekter eller regler för beteende i trafiken. Man måste ha tillräckliga kunskaper om hur bilen fungerar, hur den reagerar på olika slags agerande från föraren, man måste ha god kunskap om vägnätet, trafikregler etc., dvs en modell av det system som ska styras. Föraren måste vidare ha tillgång till alla de reglage, pedaler, ratt etc. som fordras för framförandet, dvs styrbarhet. Slutligen måste föraren ha tillgång till nödvändig information om bilens status från instrument, t. ex. hastighet, bensintillgång och temperatur, samt om bilens läge på vägbanan, omgivande trafik, position, väderlek m.m. som kan påverka agerandet, dvs observerbarhet av nödvändiga »systemtillstånd«. Om inte samtliga dessa villkor är uppfyllda omöjliggörs styrningen och uppfyllandet av huvudmålet, dvs att komma fram säkert till slutdestinationen.

I mer generaliserad form gäller, som sagt, resonemanget ovan även i sådana arbetssituationer där en dynamisk process ska påverkas/styras så att ett visst mål uppnås. Detta gäller både då man har ett mer eller mindre automatiserat styrsystem och då man har en manuell styrning eller någon kombination av detta.

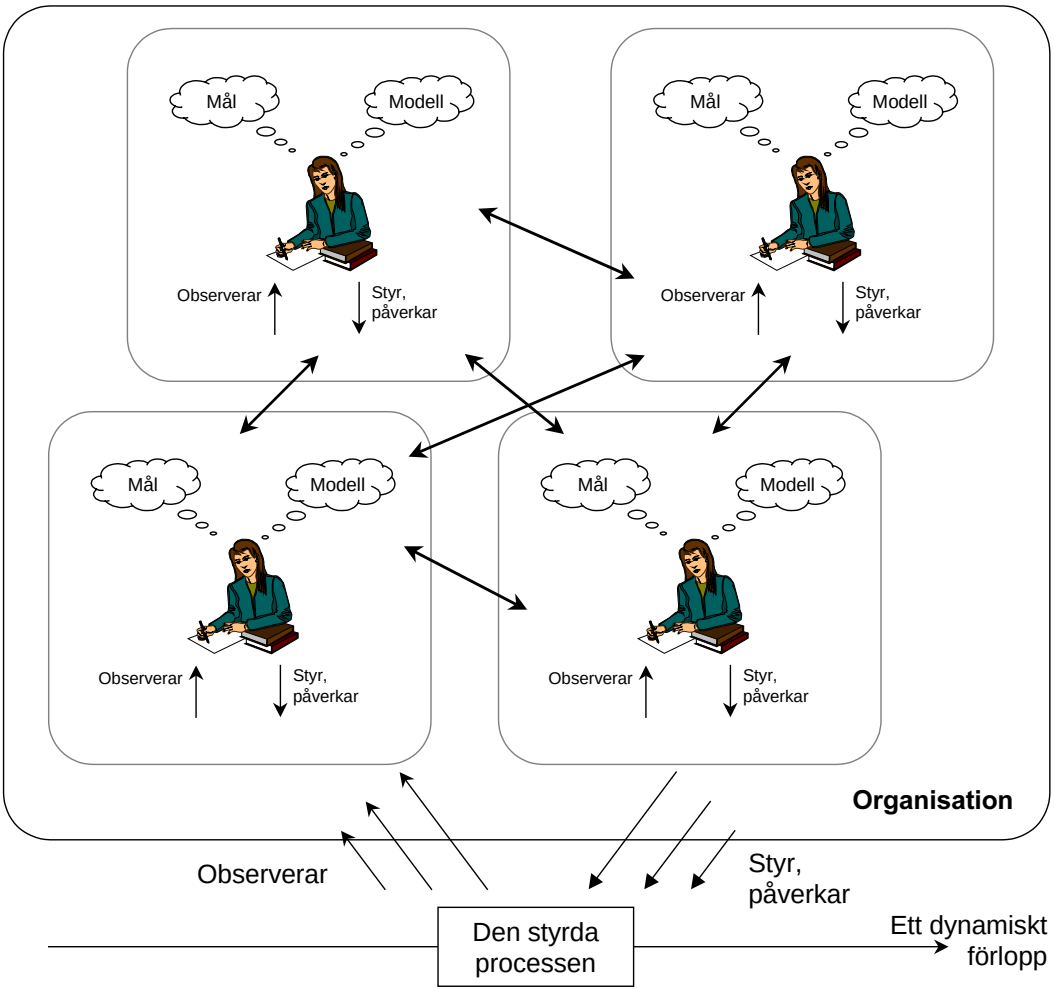
I många olika arbetssituationer gäller i praktiken dock ofta att en eller flera av förutsättningarna inte är uppfyllda. Detta leder till en rad problem. Styrandet av den dynamiska processen kan inte utföras så som det var tänkt och planerat. Verksamheten blir ineffektiv, eller mer eller mindre okontrollerad. Noggrannhets-, kvalitets- och säkerhetskrav kan inte uppfyllas. Den som ska utföra arbetet kommer dessutom att uppleva olika slags hinder, irritation, stress, hjälplöshet m.m. De problem som uppstår är alltså nära relaterade till kognitiva arbetsmiljöproblem.

I många arbetssituationer är målen inte klart formulerade, eller formulerade så att det är omöjligt för den enskilde individen att i varje läge avgöra vad som ska uppnås. Det är då dels svårt att planera och utföra sitt eget arbete på ett bra sätt, dels svårt att bedöma och utvärdera det egna eller gruppens prestationer i relation till målen. Det är heller inte ovanligt att det finns olika mål i en arbetssituation och att det finns konflikter mellan olika mål. Sådana målkonflikter kan leda till problem enligt ovan, t. ex. osäkerhet och stress, då man har svårt att veta vilka av de motstridiga målen man ska följa.

En dåligt anpassad arbetsorganisation kan också göra att man inte överblickar processen eller inte har möjlighet eller befogenhet att utföra arbetet på ett önskat sätt. Problemen kan också uppstå i sådana arbetssituationer där det finns starka inslag av delegering, antingen till medarbetare, till andra yrkeskategorier eller till ett informationssystem.

För att kunna styra ärendehanteringsprocessen på ett tillfredsställande sätt fordras således att de fyra huvudvillkoren är uppfyllda: mål, modell, styrbarhet och observerbarhet. Detta gäller för de automatiska transaktionssystemen, för den enskilde handläggaren och för organisationen som helhet.

I vår beskrivningsstruktur har vi dessutom kompletterat dessa basala villkor med ytterligare aspekter på verksamheten, för att få en ram som klarar av att beskriva det vi är ute efter. Vi måste t. ex. se olika personer med olika ansvar i planerings- och styrprocessen som samverkande. De har olika roller, ansvar, kompetenser, erfarenheter, arbetsuppgifter, mål och modeller. De kommunicerar och samverkar enligt den rådande arbetsorganisationen, samt med hjälp av tillgängliga informationssystem. Resultatet blir en mycket komplex struktur av samverkande, kommunicerande personer med olika uppgifter inom den dynamiska helheten, se figuren nedan.



Figur 4. En komplex struktur av interagerande »styrfunktioner« inom en organisation, med syfte att styra en komplex dynamisk process.

2.1.1 Beskrivningsmodellen tillämpad på ärendehandläggning

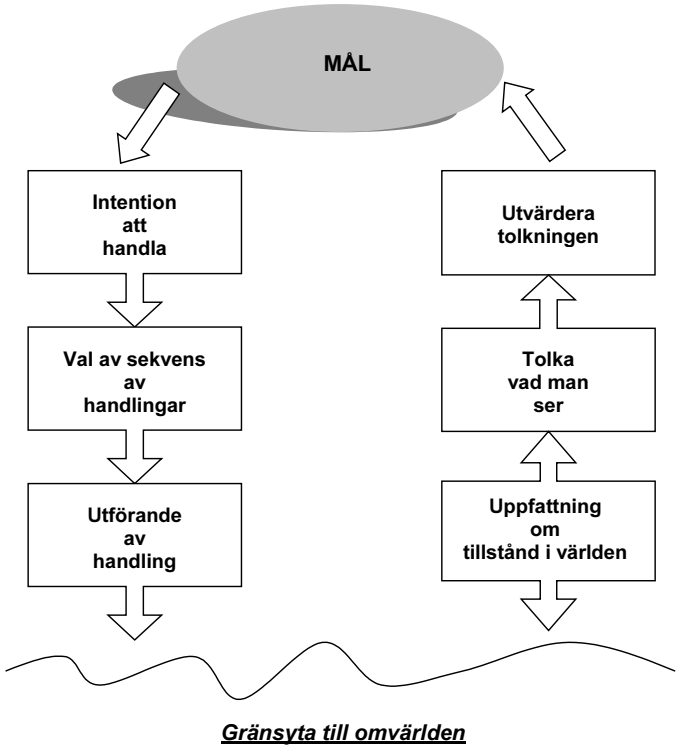
Det finns vid styrning av sådana system som det är frågan om här, egenskaper hos processen som gör att vissa av de problem som kan uppstå blir speciellt allvarliga. Ärendehandläggning är komplext, dvs det består av ett stort antal mer eller mindre autonoma delar som växelverkar med varandra. Även målen är komplexa och kan innehålla inbyggda konflikter. Systemet är dynamiskt, vilket dels innebär att tillstånden kan förändras snabbt över tiden, spontant eller som en effekt av vidtagna åtgärder, dels att beslut och åtgärder har effekt inte bara just när de utförs, utan även under lång tid framöver. Man brukar säga att systemets tillstånd och utveckling beror av dess förhistoria, eller att systemet är dynamiskt. Om man inte är medveten om dynamiken i ett system, eller har information om dess förhistoria, kan man inte fatta riktiga styrbeslut. Beslut kan ofta fattas under stark tidspress, och med högt ställda säkerhetskrav, vilket är mycket stressrelaterat. Systemet är ett tidsdiskret händelsesystem, där bara vissa

diskreta beslut kan fattas, vid vissa diskreta tidpunkter och dessutom baserat på begränsad och ofta gammal information om andra diskreta händelser. Människan har svårt att klara av många sådana beslutssituationer, t. ex. när det finns tidsglapp mellan åtgärd och information om effekten av åtgärden.

Donald Norman m. fl. hävdar att människor är målinriktade och utför sina handlingar i syfte att uppnå ett eller flera (del)mål. Själva utförandet av handlandet delar han in i två delar:

- Utföra aktionen i sig
- Utvärdera resultatet

Utförandet har sju steg, se figuren nedan [Norman, 1986]:



Figur 5. Hur vi människor utför en uppgift.

Baserat på dessa antaganden kan man dra slutsatserna att ett system och dess gränsyta måste stödja en användaren i dennes måluppfyllelse genom att:

> **Göra systemet synbart** – användaren skall lätt kunna avgöra tillståndet hos »apparaten« och få reda på möjliga handlingar.
>
> **Ha en god begreppsmodell** – designen skall återspegla begrepp som användaren förstår och kan handla efter.

Tillhandahålla lättförståeliga avbildningar – användaren måste kunna bestämma relationen mellan handling och resultat.

Ge återkoppling – användaren måste få fullständig och kontinuerlig återkoppling om resultatet av handlingarna.

Se vidare i avsnittet *2.2.1 User Centered Systems Design* om fler av Normans intressanta teorier.

2.2 Användarcentrerad systemutveckling – ett historiskt perspektiv

Ett flertal modeller existerar för hur man bör gå tillväga vid utvecklandet av interaktiva system. I detta avsnitt ämnar vi utreda hur användarcentrerade synsättet utvecklats och vilka olika trender/traditioner som i olika perioder dominerat.

Användarcentrerad utveckling innebär inte att man per automatik inkluderar användarna i processen. Vi kommer här beskriva metoder och modeller som centrerar sig på teorier om hur användarna fungerar rent kognitivt, utan att för den saken skull faktiskt involvera användarna i processen.

Användarmedverkan behöver inte vara binär, dvs. av eller på, utan kan vara flytande mellan ingen användarmedverkan alls till fullständig användarstyrning, där användarna själva driver/styr utvecklingsarbetet.

Det är viktigt att ta hänsyn till användarna i designen, att hela designen utgår ifrån deras behov, krav och förväntningar.

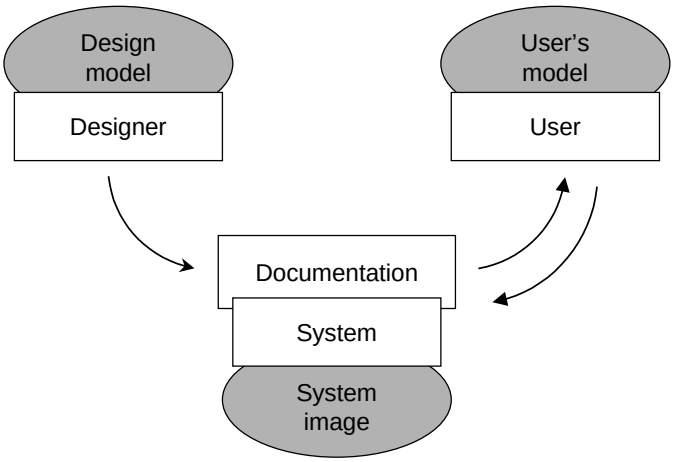
Vårt avsnitt inleds med beskrivningar av teoribaserad design och ger exempel på olika metoder inom detta område. Teoribaserad design bygger på att man som utvecklare införskaffar en stor kunskap om hur användarna fungerar rent mentalt, hur de tänker, fungerar och beter sig, etc. Man involverar inte direkt användarna i utvecklingen utan man utgår ifrån systemanalytiska teorier om att man kan representera användarna genom olika modeller som i sig imiterar människans beteende. Man använder sig av modeller av användare på vilka man gör experiment och genom att studera resultaten av dessa experiment kan man dra slutsatser om hur användaren kan förväntas reagera i olika situationer. Man måste dock vara observant på det faktum att människor är olika, att det inte går att generalisera direkt från en modell till kunskap om hur en specifik individ förväntas reagera i en specifik situation.

2.2.1 User Centered Systems Design

1986 släppte Norman & Draper en bok med titeln »User Centered System Design«. Titeln hade valts så för att förkortningen UCSD även skulle passa in på

University of California San Diego som var forskarnas hemvist. Denna bok, som är mycket läsvärd än idag, introducerade flera för tiden nya synsätt. Donald Norman beskrev i sin artikel »cognitive engineering« sin syn på problematiken med användarcentrerad systemutveckling. Normans teorier kommer från kognitionspsykologin och en av de centrala begreppen som används är mental modell.

En mental modell är användarens uppfattning av hur ett system, problem, eller dylikt hänger samman, den enklaste möjliga lösningen. T ex så kan en användare förstå hur en videobandspelare fungerar, man stoppar in ett band och trycker på `Play`. Däremot är man fullständigt ointresserad av hur informationen faktiskt lagras på bandet och hur det fungerar rent tekniskt. Användarens mentala modell är den enklast möjliga modell som hjälper till att förklara ett visst fenomen. En mental modell är unik för varje människa och ingen annan människa kan se ens mentala modell, man kan inte öppna huvudet och se hur folk tänker. Dock kan vi genom experiment och metoder se betydligt mer än man först tror.



Figur 6. System Image, den resulterande abstraktionen av designerns konceptuella modell, är vad användaren interagerar med för att bilda användarens mentala modell av systemet [Norman, 1986].

Norman [1986] definierar Design Model – den konceptuella modell över ett informationssystem som designern har, User's Model – som är den konceptuella modell som användaren formar samt System Image – som är den bild som är det synbara resultatet av den fysiska struktur som byggts (inbegripet dokumentationen och instruktioner). System Image är den fysiska bilden av en datoriserad arbetssituationen. Genom att varsebli denna bildas användarens mentala modell över systemet. Problemet med design är att skapa ett system som följer en konsekvent konceptualisering (designmodellen) så att användaren kan skapa en mental modell (användarmodellen) av systemet som står i

överensstämmelse med designmodellen. Observera att användarmodellen inte skapas från designmodellen utan från det sätt som användaren tolkar systembilden. Allt som användaren interagerar med hjälper till att forma den bilden.

Normans tankegångar var mycket viktiga för att förstå att alla användare är olika, alla användare interagerar med teknik på olika sätt och formar olika mentala modeller över hur systemet fungerar. För att förstå vad som kan orsaka användbarhetsproblem måste man alltså använda särskilda metoder för att i direkt samverkan med användarna förstå hur användarna i själva verket agerar och tänker när de använder ett visst system.

2.2.2 Modell-baserad utveckling – GOMS och KLM

Modellbaserad utveckling är den användarcentrerade utvecklingsmetod som i princip i minst utsträckning involverar användarna aktivt. Snarare är det ett tillvägagångssätt som består av djupa tolkningar av psykologiska teorier för att försöka förstå hur människorna tänker, associerar och agerar för att kunna omsätta denna kunskap till konkreta åtgärder för gränssnitten. Man bygger helt enkelt ingenjörsmässiga modeller av användare som man sedan kan använda för att förutsäga hur användarna faktiskt kommer att agera i vissa situationer. Det finns flera teorier för hur människan tänker och hur man behandlar information.

GOMS-modeller är ett sätt att bryta ned användarnas uppgifter i hanterbara delar som man kan hantera i termer av mål (Goals), operationer (Operators), metoder (methods) och urvalskriterier (Selection rules). Genom att teoretiskt analysera vad användarna vill göra när de utför sina arbetsuppgifter i termer av vilka mål de vill uppnå och hur detta gått till så kan man dra slutsatser om hur användarna faktiskt kommer att agera. Nedan följer ett exempel på en GOMS-modell över uppgiften att kopiera en artikel på en kopieringsapparat:

```
GOAL:       PHOTOCOPY PAPER
            GOAL:    LOCATE-ARTICLE
            GOAL:    PHOTOCOPY-PAGE repeat until no more pages
                          GOAL: ORIENT-PAGE
                                      OPEN-COVER
                                      SELECT-PAGE
                                      POSITION-PAGE
                                      CLOSE-COVER
                          GOAL: VERIFY-COPY
                                      LOCATE-OUT-TRAY
                                      EXAMINE-COPY
            GOAL: COLLECT-COPY
                          LOCATE-OUT-TRAY
                          REMOVE-COPY (OBS! Yttre målet fullföljt!)
      GOAL: RETRIEVE-JOURNAL
              OPEN-COVER
                          REMOVE-JOURNAL
CLOSE-COVER
```

Det intressanta som man kan notera genom denna GOMS-modell är att man fem rader från slutet faktiskt har fullföljt sin yttre målsättning, man har fått de kopior man vill ha. De deluppgifter som följer efter detta bidrar inte till att fullfölja användarnas målsättning och är därför inte viktiga för användaren. (I detta fallet kan man praktiskt notera att man flera gånger har funnit kvarglömda original i kopieringsapparater, förmodligen därför att det användarna ville uppnå var att erhålla kopian.)

GOMS-modellering är ett intressant tillvägagångssätt när syftet är att upptäcka brister i måluppfyllelsen, t ex kunde man ha förutspått det användbarhetsproblem som de initiala bankomaterna led av om man hade haft GOMS-modelleringar. De initiala bankomaterna levererade först pengar till kunden och därefter kortet (en helt naturlig uppgift eftersom in- och utmatning av kortet så att säga omsluter varandra och därför borde komma först respektive sist) vilket fick till följd att åtskilliga kunder lämnade bankomaten, nöjda efter att ha fått sina pengar, men med kortet fortfarande kvar i maskinen. En GOMS-modellering hade visat att den yttre måluppfyllelsen uppnåtts innan allt slutförts.

En annan ingenjörsmässig metod som också föreslagits av Card, Moran & Newell, (1983) kallades för KLM-metoden (Keystroke Level Model). KLM är en metod för att kvantitativt kunna predicera användarmodellen i tid. Detta gick till så att man hade tagit reda på snittider för ett antal mycket små operationer, t ex. att trycka ned en tangent på tangentbordet tar i genomsnitt x sekunder, att flytta handen från mus till styrplatta tar i snitt X sek. Denna metod skulle därför kunna användas för att avgöra huruvida en designlösning var mer effektiv än en annan, den säger dock ingenting om andra aspekter på användbarhet.

Detta tas upp inte därför att det är ett typiskt exempel på en användarcentrerad metod utan därför att det historiskt sett var intressant att studera hur användare (studenter) i själva verket inte deltog på något sätt i utvecklingsarbetet.

2.2.3 Direkt manipulering

Direktmanipulering är en av de andra kännetecknande dragen som introducerats i användarcentrerad systemdesign. Ben Shneiderman (1998) myntade begreppet för att beskriva hur användarna ges direkt fysisk återkoppling och kontinuerlig representation av sina operationer. Idag har alla grafiska miljöer (Windows, Macintosh etc) denna typ av direktmanipulation i form av exemeplvis skrivbordsmetaforen.

2.2.4 Usability Engineering

Här ser man användbarhet och design som ingenjörskonst. Med hjälp av ett förutbestämt antal designregler (heuristik/kriterier) kan man genom repetitiv design och utvärdering komma fram till en tillräckligt bra och användbar designlösning [Nielsen, 1993, Nielsen & Mack, 1994]. Usability engineering fokuserar på billiga tekniker för att förbättra användbarheten i slutet av utvecklingsarbetet, något som författarna brukar referera till som »discount usability engineering«, men som i gengäld ger omedelbara och synbara effekter direkt i utvecklingsprojekten. Eftersom de metoder som bjuds är stegvisa metoder så är de enkla att kommunicera och lära ut och enkla att använda utan nämnvärd erfarenhet i området. Metoden heuristisk utvärdering som är den mest kända metoden från detta område har på så sätt fått mycket stor spridning.

Inom usability engineering skiljer man på analytiska utvärderingsmetoder och empiriska. Analytiska metoder, som t ex de modellbaserade metoderna ovan använder sig av resonemang om hur användaren kan tänkas fungera och utnyttjar på så sätt denna kunskap i utvärderingar utan att direkt involvera användarna. Heuristisk utvärdering kan sägas vara en sådan metod. Vid empirisk utvärdering mäter man faktisk användbarhet. Detta sker genom att samla in data om hur användarna använder systemet. Ett vanligt sätt är sk användbarhetslaboratorier. Dessa är utrustade med videokameror och annan teknisk utrustning som gör det möjligt att in i minsta detalj studera vad användarna gör.

2.2.5 UTOPIA – ett praktiskt exempel på deltagande design.

Det tydligaste exemplet från deltagande design-skolan var UTOPIA-projektet som ägde rum mellan 1981–1985 men som först dokumenterades i Pelle Ehns avhandling (1988). UTOPIA studerade grafisk produktion, speciellt

bildbehandling och sidombrytning för dagstidningar, som den skulle kunna te sig med grafiska arbetsstationer med materialet direkt på skärmen. När projektet började åkte man till forskningslaboratorier som Xerox Parc för att se tekniken, mitt i projektet anskaffade man den första arbetsstationen på svenska marknaden, Perq, och projektet led mot sitt slut hade Macintoshen kommit.

I projektet deltog såväl arbetslivs- och teknikforskare från Sverige och Danmark som yrkesverksamma grafiker från de nordiska grafiska fackförbunden. I Stockholm deltog fyra av dessa användare vardera på halvtid så det fanns goda möjligheter att involvera dem i diskussioner och utvecklingsarbete. Genom utveckling av metodik för att praktiskt arbeta tillsammans, se nedan, lyckades Utopiagruppen få igång det nära samarbete på lika villkor och i lika omfattning mellan designers/utvecklare och användare som senare döpts till cooperative design eller participatory design eller den skandinaviska skolan.

I [Greenbaum & Kyng, 1991] sammanfattas och utvecklas den skandinaviska formen av kooperativ design i form av ett antal kriterier/normer:

- Datorstöd för en arbetsplats måste utformas med full medverkan från användarna, vilket kräver både specifik kompetens (som till stor del utvecklas under arbetets gång) och aktivt deltagande av användarna. Användare och utvecklare deltar på lika villkor.
- Utvecklare/designers måste ta arbetets rutiner på fullt allvar.
- Syftet med att bringa in ett datorsystem i arbetssituationen måste vara att öka kompetensen på arbetsplatsen, snarare än att rationalisera eller degradera användarna.
- Datorsystemen skall uppfattas som verktyg för att utföra ett arbete.
- Fokus skall vara att öka resultatets kvalitet likväl som produktiviteten.
- Det är viktigt att inse att designprocessen är politisk och konfliktfylld.
- Utgångspunkten måste vara i användningssituationer och rutiner i ett specifikt arbetsorganisatoriskt sammanhang.
- Arbete är att betrakta som en huvudsakligen social process.

Designprocessen baseras snarare på samarbete snarare än på formella beskrivningar. Detta innebär bl a:

- Att man utgår från användarnas rutiner för att utföra sitt arbete.
- Ett ömsesidigt lärande mellan användare och designers.

- Att man använder verktyg som är naturliga och bekanta för användaren. Lågnivå »mock-ups« snarare än formella specifikationer, t ex kartongdatorer, dia-projektorer visar skärmbilder, etc.
- Att förverkligandet av den framtida arbetssituationen tillåter användarna att experimentera med snabba prototyper i deras rätta arbetsmiljö.
- Att man måste fokusera på arbetsorganisationen minst i lika stor utsträckning som på tekniken.

I Utopia-projektet utvecklades åtskilliga lågnivåtekniker som numera generellt används för användarmedverkan och diskussion. Man använde pappkartongdatorer och skrivare, dia-projektorer som skärmarna, trä- och plastattrapper som möss, väggtidningar och läggspel för att utföra och beskriva arbetets olika faser. Mitt i projektet fick man tillgång till en grafisk arbetsstation (Perq) på vilken man i princip uppfann desktop publishing. Inom projektet utvecklades och användes en bygglåda för att visualisera alternativa arbetsorganisationer. Man spelade seriösa spel med professionella roller och kort drogs för olika situationer och åtaganden, etc. Projektet utfördes i samarbete mellan Arbetslivscentrum, KTH, universitetet i Aarhus samt de nordiska grafiska fackförbunden. Deltagarna fick omfattande erfarenhet av försök att arbeta tvärvetenskapligt.

Vad blev så resultatet? Man brukar skämtsamt säga att »operationen lyckades, men patienten dog«. Inom Utopia gjordes många viktiga upptäckter; man upptäckte hur viktigt gränssnittet är för att ett datorsystem ska bli ett bra verktyg och hur givande samarbete över ämnesgränser är. En prototyp till bildbehandlingssystem/verktyg byggdes av ett svenskt företag (Liber) till stor del baserat på Utopias specifikationer och användes i ett par pilotinstallationer i Finland och på Aftonbladet, där relativt omfattande användarstudier genomfördes. Verktyget blev aldrig en kommersiell produkt, det var inte förberett för hopkoppling med andra verktyg, t.ex. för texthantering. De erfarenheter som vanns har dock dels givit grafikerna och branschen en god framförhållning vad gäller införande av ny teknik under rimliga former vad gäller arbetskvalitet och produktkvalitet samt, som sagt, varit basen för viktig designmetodutveckling.

De experiment som utfördes med deltagande designtekniker var i stor utsträckning föregångare för nya arbetsformer och nya synsätt på systemutvecklingen som sådan, dock utförs idag inte systemutvecklingsprojekt

med deltagande design i praktiken. Det har dock varit inflytelserikt för att stärka användarnas ställning och behovet av att fokusera mer på användarnas situation.

2.2.6 Kontextbaserad design

Karen Holtzblatt & Hugh Beyer har definierat det de kallar för »contextual design« baserat på det sätt som användarna vill utföra sina arbetsuppgifter [Beyer, Holzblatt, 1998]. Denna metod innebär framför allt att systemutvecklingsverksamheten flyttar betydligt närmare användarorganisationerna och den verkliga målverksamheten.

Kontextbaserad design har följande huvuddrag:

- Designern måste förstå användarens arbetsplats, arbetsuppgifter och utifrån den förståelsen kan en prototyp tas fram, som skall testas i den riktiga kontexten – arbetsplatsen (annars är det vanligt att testa prototyper i laboratorier etc).

Man skall se på processen som ett förhållande mellan en hantverkare och en lärling. Du som studerar en professionell hantverkare är en lärling i dennes verksamhet.

2.2.7 Användarstudier

Etnografiska metoder har bland annat inom CSCW (datorstött samarbete) på senare tid växt fram som betydelsefulla analysverktyg. I etnografiska metoder är försöksledaren en del av försöket och det faktum att man är med och påverkar processen man studerar är en viktig del av förändringen man studerar. Det man egentligen skulle vilja uppnå är deltagande observationer utan intervenering. Man skall vara en naturlig del av miljön men ha andra glasögon på sig än den övriga miljön.

Användningen av videoteknik är extra viktig för etnografiska metoder såsom dokumentation av vad som sker, särskilt om det kan ske utan att störa de observerade personerna.

2.2.8 Teoretisera om användare

Att teoretisera om användarna innebär att man försöker generalisera och kategorisera användare. Genom att studera eller på annat sätt skaffa sig kunskap om användarna kan man sedan dra slutsatser om deras beteende, vilka kan ligga till grund för teorier om hur och varför användarna, generellt, gör saker och ting.

Bygger på teorier inom vetenskaper såsom kognition, argumentation, retorik, filosofi, ekonomi. Det är ett systematiskt tillvägagångssätt och kan exempelvis

användas i utvecklingsarbetet: för att prioritera; som en källa för krav; som en källa för designideer; för att fokusera utvecklingen etc.

Nackdelarna är att man kan basera för mycket av utvecklingen av ett system på teorier som är lösa antaganden eller för generella; korrekta eller felaktiga.

2.2.9 Reflektera med användare

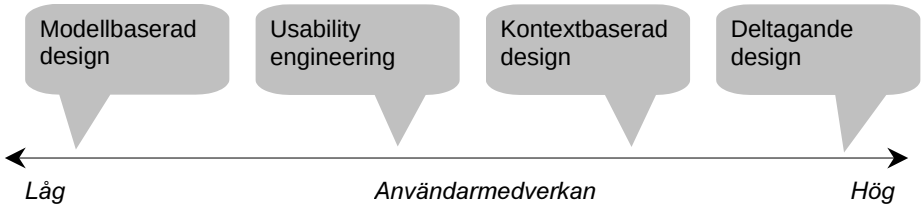
Reflektera med användarna handlar mer om att designern reflekterar över sitt eget arbete än om egentliga studier av användarna. Användarna kan ses som konversationspartners. Föregångsmannen inom denna skola är Donald Schön [Schön, 1983]. Designern kan genom att reflektera över sitt arbete hitta ny kunskap och ofta se användningsområden för sin design som den inte urspsprungligen var tänkt för. Designern ses som en forskare som utforskar nya områden med kunskap om tidigare lyckade och misslyckade designer.

2.2.10 Socioteknisk design

Socioteknisk design [Catterall, 1991] baseras på kunskapen att mänskliga och organisatoriska aspekter ej kan studeras isolerat från tekniken. Överfört till användarcentrerad design skall det finnas aktiv användarmedverkan i designprocessen, det måste vara omfattande och inte bara en del av en utvärderingsprocess i slutet av utvecklingen.

2.2.11 Sammanfattning

I ovanstående avsnitt har vi bara mycket kort redogjort för ett axplock av de modeller som finns. De olika synsätten på den användarcentrerade designprocessen kan ses ur perspektivet av olika grader av användarmedverkan:



Figur 7. »Grad« av användarmedverkan vid olika synsätt på designprocessen.

Vi har lagt in fyra exempel i figuren ovan. Detta för att illustrera att de olika synsätten, eller tillvägagångssätten hanterar användarcentreringen och användarmedverkan på olika sätt och bör beaktas utifrån sina för- och nackdelar.

2.3 Tänkbara problem vid införande av användarcentrering

Arbete med datorstöd kan i många lägen medföra flera olägenheter bland annat, pga bundenheten vid datorstödet, något som grundas mycket tidigt i utvecklingsarbetet. Det finns dock en stor mängd andra problem för användarna och många av dessa kan direkt hänföras till datorstödet. Hur skall man då kunna undvika dessa problem? Eftersom problemen ofta grundas tidigt i utvecklingsarbetet är det viktigt att studeras utvecklingsprocessen, hur såväl IT-stöd, organisation, verksamhet och kompetens utvecklas i ett helhetsperspektiv. I det följande avsnittet skall vi beskriva några av de problem som vi stött på inom de projekt vi varit inblandade i.

2.3.1 Problem vid systemutvecklingsarbete

Det finns en stor mängd problem som blir uppenbara i systemutvecklingsprojekt och som försvårar effektiv användarmedverkan. Nedan listas dessa problem och diskuteras i relation till RSVs verksamhet:

- **Bristande tid.** Det största hotet mot att utföra användbarhetsrelaterade insatser är bristande tid. Projekten på t ex RSV som vi har deltagit i upp igenom åren har initialt kännetecknats av att man har oändligt mycket tid till sitt förfogande och att man därför skall spara användbarhetsrelaterade insatser till senare i projektet. Så vid en viss tidpunkt i projektet blir det kris och man förstår inte hur man skall hinna med att få fram ett fungerande system. Mycket tid och kraft går åt till att fundera över hur man skall lösa de akuta problemen och hur man överhuvudtaget skall få fram ett fungerande system – användbarhet blir av sekundär betydelse. Detta är ofta i projektets konstruktionsfas som projektet tenderar att försenas, när det blir uppenbart vilka problem som faktiskt står att lösa. Det finns en vida spridd missuppfattning att iterativt arbete skulle ta mer tid i anspråk än traditionellt utvecklingsarbete. Vi har varit med om exempel där man enbart har hunnit med utvärderingar av användbarheten i projektets absoluta slutfas – utan att det fanns någon tid att korrigera de problem som då upptäcks. Det har t o m funnits exempel på att projektledare hört av sig till utvärderare och påpekat vikten av att användbarhetsutvärderingen visat positiva resultat, eftersom detta starkt kommer att påverka attityderna hos de som skall använda produkten. Paradoxalt nog är de problem som uppstår till följd av bristande tid huvudsakligen en produkt av att utvecklingsprojekten är alldeles för långa. Det är mycket större risk för tidsbrist i ett sjuårigt projekt än i ett 3-månaders projekt.
- **Gör mindre projekt.** Ju större och längre projekt desto större sannolikhet för att projektet misslyckas, avbryts eller kommer att uppvisa allvarliga

brister. Ett sätt att uppnå detta kan vara genom att dela in projekten i små delar som producerar s k deliverables, t ex genom att man producerar, implementerar och testar en arbetsyta i taget. För att detta skall vara möjligt bör man ha en modulbaserad struktur av projekten i vilken varje separat del kan fås att fungera och sammanfogas till en fungerande helhet.

- **Det är bråttom från första början.** Agera som om det var väldigt bråttom från första början. Satsa på att komma fram till de saker som kan ha betydelse för användarna på ett tidigt stadium, dvs. skapa prototyper av användarytorna tidigt.
- **Användbarhet tidigt in i projektplaneringen.** Satsa på att direkt från början ha en användbarhetssyn på utvecklingsprojektet, så att man kan uppnå och utföra användbarhetsrelaterade insatser direkt från början. På så sätt kan man låta användbarhetskompetensen komma in som ett stöd på projektledningsnivå snarare än som små isolerade öar sent i projektet, utifall det finns tid kvar.
- **Attitydproblem.** Vilken attityd systemutvecklare har till utvecklingsarbetet påverkar starkt resultatet. Systemutvecklare har svårt att se sin arbetsroll som service i ett arbetssammanhang. Många systemutvecklare ser sin roll som att bryta nya marker å det tekniska perspektivet, lösa tekniska problem, och blir irriterade över användarnas naiva kommentarer eftersom de inte »ser det fiffiga i programkoden«.
- **Kommunikationsproblem.** Bristande förmåga, tid eller intresse att förstå och tolka de respektive olika världar som är en del av de människor som ingår i systemutvecklingsarbetet är i botten ett problem i kommunikationen mellan människor. Inom RSVs projekt accentueras detta särskilt. Den genomsnittlige handläggaren (systemanvändaren) är kvinna i 50-årsåldern emedan den genomsnittlige systemutvecklaren är en man runt 30 år. De kulturella skillnaderna i termer av kompetens, livserfarenhet och intresse är givetvis enormt stora för dessa olika grupper. Att få tillstånd effektiv kommunikation mellan dessa olika parter är ett stort problem. Men vari består problemen? Är det grupprocessen som är problematisk eller det sätt på vilket deltagarna samarbetar? Begär vi för mycket av användarna? Pratar de olika deltagarna ens en gång samma språk? Använder någon i projektet sin makt för att förhindra att någon deltagares synsätt inte kommer i dagen?
- **Terminologiproblem.** Att kommunicera med begrepp som är bekanta för användaren är viktigt för att kunna få dessa engagerade i utvecklingsarbetet. Användarna bör inte huvudsakligen arbeta i termer av teknik. De skall inte behöva uttrycka sitt arbete i termer av fönster och ikoner utan snarare i

termer av sina arbetsuppgifter. Att få till stånd en användarcentrerad utvecklingsterminologi är viktigt för att få användarna att känna sig delaktiga. En kravspecifikation uttryckt i användarnas terminologi kan behöva översättas för att skapa en kravspecifikation som är användbar för utvecklarna. Det är viktigt för utvecklarna att lära sig att använda användarnas terminologi.

- **Metod och verktygsproblem.** Används rätt metoder för rätt saker? Är befintliga metoder och verktyg för systemutveckling användbara för alla som medverkar i processen? Att lära sig att bemästra hur man gör en begreppsmodellering eller en processmodellering kan ofta vara en tuff process för en användarrepresentant och något som kan upplevas som ett meningslöst pussel. Stödjer de metoder som amvänds ett användarcentrerat synsätt, stödjer de det kreativa arbetet i att skapa ett nytt arbetssätt för användarna? På denna front finns mycket övrigt att önska. Tag som ett exempel metoden för gränssnittsmodellering som tagits fram i samverkan med CMD. Så som denna metod har kommit att användas på senare tid så har den mer kommit att bli »ytterligare en modellering« snarare än ett stöd för att därefter kunna utföra gränssnittsdesign som är anpassat till arbetsuppgiften.
- **Organisationsproblem.** *En* förutsättning för att kunna göra användbara system är stöd och uppmuntran från hela den inblandade organisationen. Otillbörligt ledningsinflytande, makt och konfliktsituationer mellan projektdeltagare, obstruktion, gruppindelning etc. är bara några exempel på de problem som man brukar finna i dessa sammanhang. Det är t ex ovanligt att projektledningen allokerar tillräckligt med tid och resurser för användarcentrering i utvecklingsprocessen.
- **Deltagarnas stöd.** Det måste finnas stöd för användarcentrerat arbete inte bara formellt genom organisation och budgeteringsaspekter utan även genom medlemmarnas moraliska stöd. Detta är en förutsättning för att man skall kunna känna att man kan delta i den utsträckning som är nödvändigt. Det räcker i princip med att en av medlemmarna i projektet motarbetar synsättet så kommer det inte att bli fruktbart. Användarnas stöd kan uppnås genom relativt enkla medel:

    1. Låt alltid alla komma till tals och att alla synpunkter kommer fram (även från de som inte spontant uttrycker så mycket).
    2. Se till att dokumentera alla användarsynpunkter fortlöpande, se även till att användarna delges resultatet av det beslut som fattas

till följd av användarnas synpunkter även om det inte är positivt för dem.

3. Undvik »my-baby-syndromet«, dvs. att någon enskild fått arbeta så mycket på ett visst resultat att man inte är mogen att ta in synpunkter eller förändringar utan snarare kommer att försvara en ståndpunkt.

- **Kompetensproblem.** De personer som deltar i designprocessen har sällan kunskap, erfarenheter eller de speciella förmågor eller ens intresse som krävs för att arbeta användarcentrerat. Människa-datorkunskap är trots årationden av forskning fortfarande relativt okänt för de flesta som deltar i systemutvecklingsarbete. Här har vi som forskare en viktig uppgift i att sprida och göra denna kunskap tillgänglig för allmänheten. Med kunskapen följer också förändrade attityder till arbetssättet.
- **Externa aspekter.** Allting kan hända i verkliga projekt. Användarcentrerat arbete utförs aldrig isolerat från projektet, tvärtom kan en mängd oväntade aspekter störa utvecklingsarbetet. Ett projekt kan påverkas av förändringsarbete i organisationen, politiska eller strategiska beslut och dessa beslut är sällan enkla att påverka. Ofta finns olika intressen representerade i projekten och de konflikter som dyker upp kan kanske inte alltid lösas inom ramen för projekten.

Trots alla de olika problemområden som vi identifierat så är användarcentrerad systemutveckling någonting som vi ser som en nödvändig förutsättning för att åstadkomma användbara system. Utan att blanda in användarna i systemutvecklingsarbetet kan vi aldrig någonsin göra system som blir användbara för den faktiska slutanvändaren. Vi har valt att ta upp dessa olika problemområden därför att om de synliggörs, kan man aktivt arbeta för att komma undan dessa problem.

3

Några metodsteg

En användarcentrerad systemutvecklingsmodell kräver bl. a. delvis nya metoder. I detta kapitel redogör vi för några av de vi tycker är viktiga, och vilka till stora delar saknas idag inom de organisationer vi har studerat.

3.1 Användar- och uppgiftsanalys

En systemutvecklare är inte den typiske slutanvändaren av de system eller tillämpningar som vi utvecklar. Om vi inte lyckas förstå vilka arbetsuppgifter som användarna utför eller hur de utför dem, kan vi inte heller utveckla ett system eller en produkt som möter deras krav och förväntningar. Att förstå de tilltänkta användarna, deras arbetsuppgifter, arbetsmiljö, organisation etc. är en nyckel till hela utvecklingsprocessen. Vi kan samla in högvis med data, analysera, köra stora tester, men om deltagarna inte är representativa för slutanvändarna eller om vi inte lyckats förstå hur och varför arbetsuppgifter utförs, vad har vi då tjänat? Kunskap om användarna inkluderar saker såsom deras kunskapsnivå, utbildning och träning, användningsfrekvens, arbetsmiljö, arbetsuppgifter m.m.

Hur kan vi samla in denna kunskap om användarna? Det absolut bästa sättet är att studera, intervjua och *leva* med användarna i deras arbetsmiljö. Detta ger

nödvändiga och oersättliga insikter i arbetsförhållanden, användargrupper och andra förutsättningar. Vi måste ut till användarnas arbetsplats för att lyckas fånga saker som inte går att fånga på annat sätt än genom fysisk närvaro. Det är först då vi har möjlighet att se informella organisationsstrukturer, ta del av »tyst kunskap« etc. Om detta av olika skäl inte är möjligt kan man fånga delar av den nödvändiga kunskapen genom t.ex. marknadsundersökningar eller enkäter. Resursåtgången och kostnaden för en analys av användarna kan variera oerhört, allt ifrån det mycket spartanska och billiga, till det mer genomgripande och därmed kostsammare. Kom ihåg att det kan finnas en hel del information tillgänglig i sammanhang som kanske inte är uppenbara. Det kanske finns en tidigare gjord analys, branschorganisationer kan ha bakgrundsmaterial etc.

Syftet med användar- och uppgiftsanalyserna är att tillsammans med användarna ta reda på vilka arbetsuppgifter som utförs, vilka som utför arbetsuppgifterna, hur arbetsuppgifterna utförs, hur nuvarande stödsystem fungerar och hur användarnas arbetssituation och informationsstöd kan förbättras.

ANVÄNDARANALYSEN skall svara på frågor om vilka användargrupper som finns i den organisation för vilken ett stödsystem skall utvecklas, samt vilka egenskaper dessa grupper har. Skillnader mellan olika grupper av användare är mycket viktiga att ta hänsyn till vid design av det framtida stödsystemet. Lättidentifierade grupper av användare är:

- Experter – nybörjare
- Gamla – unga
- Män – kvinnor
- Vana datoranvändare – ovana datoranvändare
- Liten verksamhetskunskap – stor verksamhetskunskap

Andra intressanta användaregenskaper av vikt är:

- Erfarenhet av arbetsuppgifterna de skall lösa. Är det huvudsakligen nyanställda som har liten vana av att klara av arbetsuppgifterna eller har företaget en personalstyrka som allihop skött arbetet på ett visst sätt? Det inses lätt av båda fallen sätter speciella krav på användargränssnittet.
- Vilken utbildning har användare – både akademisk och mer företagsanpassad sådan. Vilka språk behärskar de, är ett program där engelska termer förekommer ett problem?
- Vilken datorvana har användarna. Har stor datorvana och har använt de senaste typerna av datorsystem, eller har de kanske ingen datorvana alls?

- Hur mycket tid kommer läggas på utbildning?
- I vilken miljö kommer systemet att finnas? Kommer användarna att sitta i kontorslandskap eller i enskilda rum?
- Finns det amvändare med speciella fysiska hinder?

Man kan ta reda på ovanstående information genom en rad olika metoder eller tillvägagångssätt, exempel är:

- Enkäter
- Intervjuer
- Observationer

Oavsett metod syftar denna analys till att urskilja typer eller grupper av användare med liknande bakgrund, förutsättningar, arbetsuppgifter och krav på användargränssnitt. Analysen resulterar i; användarprofiler, designrekommendationer (eller kriterier) samt delar i ett underlag för exempelvis en kravspecifikation.

UPPGIFTSANALYSEN skall svara på frågor om vilka arbetsuppgifter användarna utför och hur dessa genomförs.

Frågor som denna analys skall ge svar på är av typen:

1. Varför utför användaren en viss arbetsuppgift?
2. Hur ofta utförs arbetsuppgiften?
3. Hur lång tid tar det att utföra arbetsuppgiften?
4. Vilka steg eller handgrepp behövs för att utföra arbetsuppgiften?
5. Samarbetar användaren med någon annan när arbetsuppgiften utförs?
6. Vilka hjälpmedel; system, produkter, blanketter etc, behöver användaren för att utföra arbetsuppgiften?
7. Finns det mycket kritiska arbetsuppgifter och »flaskhalsar«, vilket gör arbetsuppgiften svår att utföra?
8. Hur kan arbetssituationen och informationsstödet förbättras?

För att ta fram ett användbart system är det självklart viktigt att göra en noggrann arbetsuppgiftsanalys (eng. *task analysis*) – vilka funktioner skall systemet kunna utföra. Om inte systemet klarar av att lösa de relevanta problemen kommer det ju inte heller att användas. Problemet är bara att allt för ofta implementeras allt för mycket funktionalitet – med följd att användarna får svårt att hitta de relevanta funktionera. Faktum är att en ordentlig analys av

vilka uppgifter som skall lösas i arbetet kan ibland radikalt minska storleken på system och minska komplexiteten.

Uppgiftsanalysen kan genomföras med olika metoder. De vanligaste är strukturerade intervjuer och observationsintervjuer. Bra att tänka på vid intervjuer kan vara:

- Fråga om specifika och konkreta situationer och produkter av användarnas arbete.
- Undvik abstrakta frågeställningar om hur de tror deras arbetssituation kan förbättras och ändras.
- Komplettera med observationer av användare i arbete.

Exempel på frågor att ställa vid intervjuer:

- »Varför gör du det?« – relatera till mål.
- »Hur gör du det?« – dela in i deluppgifter.
- »Varför gör du inte så här?«
- »Blir det ibland fel när du gör det här?«
- »Vad händer om det blir fel?«

Förslag på arbetssätt vid beskrivning av resultaten från uppgiftsanalysen:

- Formulera mål och delmål.
- Formulera alltid ett övergripande mål med hela interaktionen.
- Formulera metoder för att uppnå målen.
- Formulera vilken information som behövs för att uppnå målen.
- Gör uppdelningen »bredden först« – identifiera gemensamma delmetoder.
- Stanna uppdelningen då »löven« i form av arbetsmoment nåtts.
- Arbeta gärna med papper och penna t.ex. Post-It Notes.

Resultatet blir ett material som tillsammans med användaranalysen bildar en bra utgångspunkt och underlag för designprocessen.

Det är viktigt att man i samband med uppgiftsanalysen börjar formulera användbarhetsmål. Dessa mål kan formuleras med utgångspunkt i ISO 9241-11:

- Effektivitet – måluppfyllelse
- Produktivitet – måluppfyllelse i förhållande till resursåtgång
- Acceptans – användarnas tillfredsställelse

Dessa användbarhetsmål är direkt nödvändiga om den iterativa processen skall komma till ett slut. Målen indikerar om processen går åt rätt håll – »om du inte kan mäta så kan du inte styra«. Användbarhetsmålen kan formuleras i termer av:

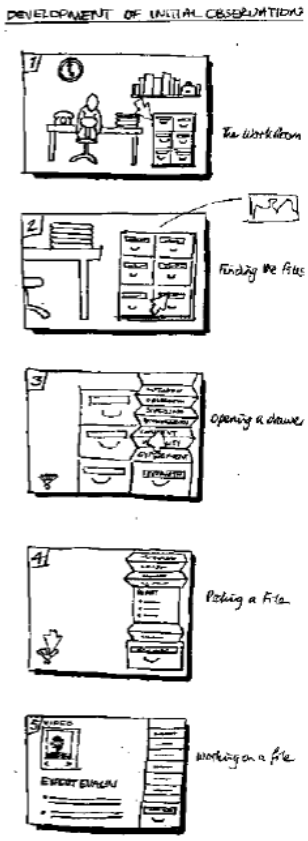
- Inlärningstid.
- Tid för att lösa uppgifter.
- Antal fel som användare gör.
- Hågkomst över tiden.
- Användarnas uppskattning av programmet (»nöjda användare«).

Ett exempel kan vara:

> *En användare bör efter 2 timmars träning kunna klara av att mata in minst 10 formulär och under tiden göra högst två fel.*

3.2 Scenario och storyboard

Scenarier och storyboard är exempel på tekniker för att utforska och visualisera olika tänkbara designlösningar (se exempelvis referensen: Löwgren & Stolterman, 1998, sidorna 123–131). I scenarierna beskriver man hur det är tänkt att systemet skall uppföra sig och hur olika arbetsuppgifter skall gå att lösa med hjälp av systemet. Man tar alltid fram mer än ett scenario. Varje scenario skall beskriva hur en viss kategori användare utför viss eller vissa arbetsuppgifter med hjälp av systemet. De skrivna scenarierna kan kompletteras med skisser och bilda storyboards där användargränssnittets olika utseende och uppförande beskrivs bildligt likt en filmsekvens. Arbetet med scenarier och storyboards görs oftast med hjälp av papper och penna och resulterar i tidiga prototyper av systemet. Exempel på ett storyboard:



Figur 8. Exempel på storyboard

Det är ovärderligt att kunna visualisera olika förslag på designlösningar och diskutera dessa med användarna. Det går även utmärkt att genomföra utvärderingar av förslagen, och på så sätt tidigt i utvecklingsprocessen mäta mot uppställda användbarhetsmål.

3.3 Design och utformning

I dagens utveckling av användargränssnitt separeras oftast innehållet och presentationsformen. Så kallade Style Guides och riktlinjer (exempelvis *The Windows Interface Guidelines for Software Design* [Microsoft, 1995]) tillämpas regelmässigt och utan någon större tanke på vilken form eller innebörd informationsinnehållet har. Om man bara följer dessa riktlinjer utformas t ex alla informationsfält i användargränssnittet likadant, oavsett om de skall representera ett namn eller en busstid. Detta är vare sig önskvärt eller bra. Istället behövs en mer genomtänkt och effektiv *design* av användargränssnitt.

I sin bok Visual Function [Mijksenaar, 1997] argumenterar Paul Mijksenaar (professor på Delft University of Technology, Holland) för att design består av

tre oskiljbara delar; *durability*, *usefulness* och *beauty*. Dessa tre komponenter kan blandas till olika delar för att passa den aktuella tillämpningen:

– Design is an activity that unities the elements of durability and usefulness and intensifies the perception of beauty.

Design after all has the unique capacity to shape information by

- *emphazising or understating,*
- *comparing or ordering,*
- *grouping or sorting,*
- *selecting or omitting,*
- *opting for immediate or delayed recognition,*
- *presenting it in an entertaining fashion.*

Naturligtvis skall man försöka att utnyttja de möjligheter som design ger. Vi kan dock konstatera att designmomentet till stora delar saknas i dagens arbete med användargränssnitt. Det finns förstås ett och annat undantag. Spelprogram, program för barn etc är ofta mycket väl designade, men är fokuserade på underhållning och inte avsedda att vara produktiva eller effektiva för en viss arbetsuppgift. Ett växande område för design är marknadskommunikation bl a via webben. Dessa försök är i och för sig intressanta men faller oftast på bristande användbarhet. Alltför mycket fokuseras på att presentera »flashiga« bilder och användbarheten kommer i skymundan. Hur viktig är då design och estetik i ett användargränssnitt? Inom mer mogna designområden såsom industriell design och arkitektur har man länge jobbat mycket medvetet med design som ett sätt att skapa en effektiv arbetsmiljö på. Där har man lyckats förena form och funktion via skicklig design. I datorvärlden (åtminstone den administrativa) ser i stort sett alla användargränssnitt likadana ut dvs som Windows. Med det menar jag att designen som ett självändamål ser likadan ut från program till program. Windows som begrepp och valt utvecklingsverktyg styr helt och hållet designen. Designen uppstår mer än den är medveten. Användargränssnittet får en design som speglar utvecklingsverktyget istället för den verksamhet i vilken systemet skall användas. Man kan nästan se vilket utvecklingsverktyg som använts bara genom att titta på ett användargränssnitt. Detta är lika tokigt som om din frisör skulle plocka fram kökssaxen när han eller hon skall klippa dig. Bara för att det är en sax är den inte identisk med alla andra saxar. Frisörens sax är t ex gjord i en speciell legering (bl a med kobolt för att få rätta känslan), är slipad på ett speciellt sätt, har ett visst tumgrepp etc. En frisör kan inte utföra sitt jobb effektivt med en sax som inte har rätt egenskaper, såväl funktionella som designmässiga. Precis som saxen måste användargränssnittet anpassas till den miljö det skall fungera i. De subjektiva

värderingarna och estetiken är faktorer som påverkar helheten, vilket i sin tur ökar *användbarheten*.

Hur skapas en effektiv design? Många försök har gjorts att beskriva begreppet design inom systemutvecklingen. Jag anser att man bör se design som en dynamisk process och både som en konstart och en vetenskap. Den kanske bästa och vetenskapligt grundade beskrivningen av design som beteckning på hela utvecklingsprocessen står Carroll och Rosson (1985) för. De föreslår fyra aspekter vilka karaktäriserar design:

1. Design is a process, it is not a state and cannot be adequately represented statically.
2. The design process is non-hierarchical, neither strictly bottom-up nor top-down.
3. The process is radically transformational, involving the development of partial and interim solutions which may ultimately play no role in the final design.
4. Design intrinsically involves the discovery of new goals.

Designen av ett system pågår alltså under hela systemutvecklingen – systemet specificeras inte bara för att sedan implementeras. Denna överordnade beskrivning av design som utvecklingsprocess leder sedan vidare till beskrivningar av design för specifika faser inom utvecklingsprocessen. Vi är speciellt intresserade av design i syfte att skapa ett effektivt och användbart system – design av användargränssnitt. I detta sammanhang kan vi luta oss mot fem kärnaktiviteter beskrivna av Crampton Smith och Tabor (1996):

1.

Förstå. Vad är det som pågår? Vilka problem är det som skall lösas? Designern studerar och pratar med användare och intressenter – analyserar. Den studerade verksamheten dokumenteras med bilder, anteckningar, video, strukturerade intervjuer etc.

2.

Abstrahera. Vilken är huvuddelarna? Vilken typ av information används? Hur används informationen? Vad är viktigt? Vad är oviktigt? Skisser, diagram, listor etc används som dokumentation.

3.

Strukturera. Hur förhåller sig de olika delarna, till varandra? Hur kan delarna struktureras för att bli

effektiva för användarna? Vad är användarna intresserade av? Designers bild av strukturen stäms av med användarnas bild.

4.

Representera. Hur kan denna struktur representeras? Finns det någon »inneboende« representation hos informationen? Hur kan designern representera informationen med tanke på användarnas bild? Skall representationen vara konkret eller abstrakt? Kan en metafor användas? Designern jobbar i detta skede med skisser på papper och bildspel samt med interaktiva prototyper.

5.

Detaljera. Vilka attribut skall användas? Hur skall detaljerna utformas? Vilken stil skall utseendet formas efter? Hur utformas dynamiken? Kanske skall en illustratör anlitas.

Dessa fem aktiviteter utförs nödvändigtvis inte i sekvens. Allt eftersom arbetet fortskrider växlar designern mellan dem. En aktivitet påverkar de andra och designern får ständigt cirkulera sina arbetsuppgifter.

3.4 Estetisk design

Att det datorstöd man arbetar med upplevs som estetiskt tilltalande är onekligen en aspekt som upplevs som allt viktigare i de flesta situationer. Det är dock omtvistat vad man menar med estetiskt tilltalande. Helt klart är att det är en subjektiv uppfattning av ett systems utseende vad gäller färg, form, layout, typografi, etc. Även om ingen absolut sanning finns så känner vi till en hel del om hur människor tenderar att uppleva färger och färgkontraster, harmoniska proportioner, linjeringar, stil, typsnitt, etc.

Den estetiska designprocessen är ett område som är betydligt mindre utforskat. De människor som är inblandade i skapande arbete (grafiska designers, industridesigners, arkitekter, målare, etc.) har ofta svårt att beskriva hur den skapande processen går till. Därför är det också svårt att hämta generella tumregler eller riktlinjer som man kan följa om man inte har denna kompetensprofil.

Helt klart är det estetiska designarbetet att betrakta som ett hantverk, resultatet bara uppstår, utan att man i efterhand kan sätta fingret på hur det har gått till.

Precis som i de flesta estetiska utbildningar så är den estetiska arbetsprocessen präglad av **erfarenhet** och **talang**. Man lär sig att bli en bra designer genom att göra mycket design, och gradvis förfinar man sina redskap och metoder. Den estetiska kompetensen (t ex grafisk designer, industridesigner eller informationsdesignern) är än idag ganska sällsynt i systemutvecklingsarbetet, men i allt större utsträckning börjar man idag anlita sådan personal, fortfarande mest på konsultbasis. Det borde vara möjligt att bygga upp denna typ av kompetens inom stora utvecklande organisationer, men detta kan bara ske under förutsättning att designern får ägna sig åt designuppgifter och inte drunkna i annat arbete. På så sätt kan designern bidraga till att gränssnitt från olika delar av organisationen får samma »stil«.

Det är viktigt att vi inte börjar för tidigt med det estetiska designarbetet. Att »brainstorma« olika designlösningar kan försvåras om designen ser alltför färdig ut. Dock borde estetisk design kunna ske på två olika nivåer. Dels genom att grundläggande element ges en genomtänkt design såsom generella återanvändbara moduler samt genom att den estetiske designern lägger sista handen vid designen i termer av proportioner, form etc.

Huruvida man skall använda sig av en grafisk designer eller en industridesigner beror ganska mycket av vilket resultat man vill eftersträva. En grafisk designer kan vara fokuserad på mjukvaruinnehållet och utseendet på skärmen, emedan en industridesigner kanske hellre fokuserar på hårdvaruaspekter och kanske mera arbetar med prototyperna än med den slutliga produkten.

Den estetiske designerns roll i utvecklingsarbetet är på inget sätt solklar, eftersom formerna för denna typ av verksamhet inte riktigt satt sig ännu. Det är också troligt att en grafisk designer har en annan uppfattning en den vi beskriver av hur denne tycker att arbetet med estetisk design skall bedrivas. Var, när och hur sådant arbete skall beskrivas måste bli föremål för utredning och borde specificeras i organisationens utvecklingsmodell.

3.5 Prototyping

Ibland uppstår lite förvirring när man talar om prototyping som en teknik eller metod vid systemutveckling. Detta bottnar bl. a. i att andra discipliner, såsom industriell design, sedan länge har definierat prototyp som den första färdiga produkten att sätta i produktion (Norstedts engelsk-svenska lexikon, andra upplagan 1993: ***Prototype:*** *prototyp, första modell; urtyp, urbild*). Innan dess har industridesignern tagit fram en eller flera »mock-ups« och skisser. Själva processen för att ta fram dessa artefacts (mock-up, skiss och prototyp) är design. Vi inom systemutvecklingsvärlden bör vara lite varsamma med hur vi använder begreppen prototyp, prototyping och design.

Vidare är viktigt att inse att vi inom systemutvecklingsvärlden inte fullt ut kan se på en prototyp på samma sätt som en industridesigner, då ett datorprogram inte har känslan av fysiskt material på samma sätt som t ex en prototyp för en bil. I brist på bättre ord eller standarder använder vi i denna rapport begreppet prototyp som beteckning på ett system innan det är helt färdigutvecklat och prototyping som benämning på en teknik att få fram prototyperna på.

Prototyping som en metod eller teknik vid systemutvecklingen ger oss bl. a. följande möjligheter:

- Utforska nya lösningar
- Prova funktionalitet
- Hitta krav
- Träna kreativitet; möjligt att lära sig och att bli bättre
- Hitta svagheter
- Testa prestanda, utseende etc
- Prova sekvenser av kommandon
- Ett sätt att utveckla system på!

De finns en rad olika synsätt på prototyping; vad det är och vad det inte är. Det finns en skola representerad av bl. a. Budde [Budde, 1992] som anser att man kan se prototyping som ett sätt att utveckla sitt system eller produkt på:

– »Prototyping is an approach to software development incorperating the following features:

  - Operative versions are produced at an early stage
  - Relevant problems are clarified by experimentation
  - Prototypes provide a common basis for discussion between developers, users and other groups«

– »Prototyping has grown out of the realization that:

  - requirements frequently do not become apparent until a system is in use
  - specifications cannot be completed until during the construction process
  - users and developers must learn from each other«

Andra skolor hävdar att prototyping enbart leder till låg kvalitet på produkten. Förespråkare av den senare skolan är dock oftast utvecklare vilka utvecklar

system av mycket teknisk art där interaktionen med en användare är av mycket underordnad betydelse. Vikten av prototyping som ett sätt att finna lösningar på är hur som helst vedertagen.

3.5.1 Metoder för prototyping

Tittar man i en klassisk skolbok för människa-datorinteraktion hittar man följande varianter av prototyping [Preece et. al., 1994]:

- *Requirements animation:* olika möjligheter demonstreras.
- *Rapid prototyping; »throw-it-away«:* samla in krav eller prova designlösningar, utvärdera och släng bort.
- *Incremental prototyping:* systemet byggs inkrementellt.
- *Evolutionary prototyping:* kompromiss mellan produkt och prototyp, systemet utvecklas och förfinas under hela utvecklingsprocessen.

Om vi generaliserar en aning kan vi slå ihop requirements animation och rapid prototyping samt incremental och evolutionary prototyping till två huvuddelar:

3.5.2 Rapid prototyping

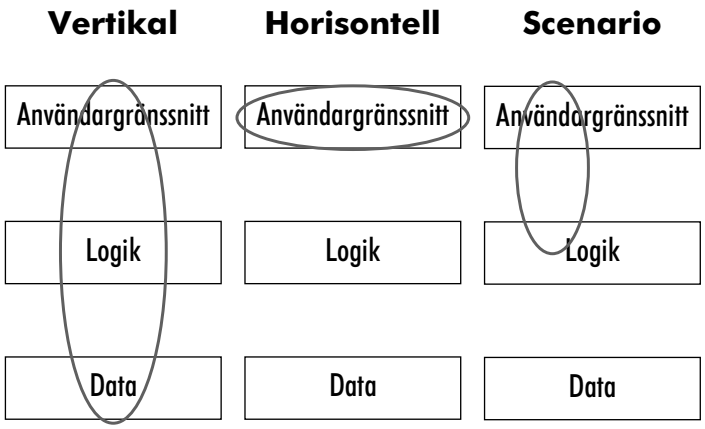
- »Throw-it-away« – slängs bort efter användning.
- Måste vara billig efterson den slängs bort.
- Prototypverktyg används; högnivåverktyg, schematiska skissverktyg.
- Samlar in ny information om systemkrav – är mest utforskande av alla tekniker.
- Testar designlösningar.
- Provar om lösningar är rimliga.
- Används för att evaluera saker, innan man väljer inriktning, poängen är att man har tid/råd att göra många olika lösningar.

3.5.3 Evolutionary prototyping

- En slags kompromiss mellan »riktig utveckling« och prototyping.
- Bygger systemet gradvis, bildar till slut det riktiga systemet.
- Försöker överbrygga gapet mellan prototyping och implementering.
- Viktigt att välja det slutliga verktyget.

- En svaghet är att man gärna fixar brister och fel istället för att utvärdera alternativ – »my baby«.

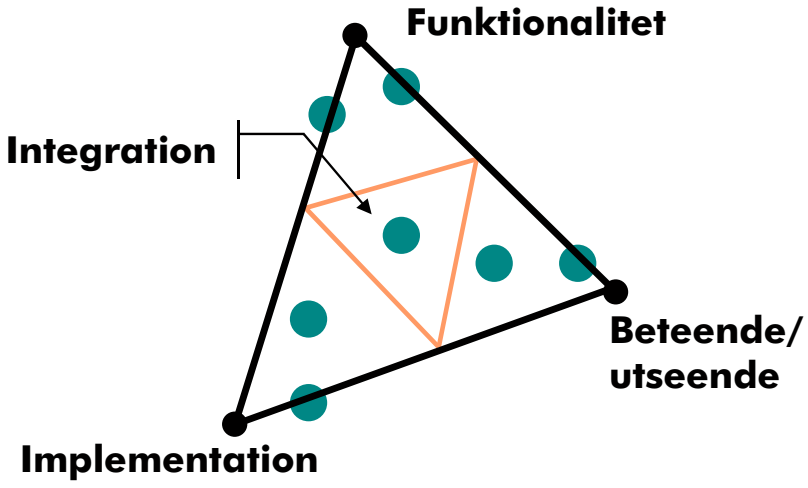
Man kan ha olika fokus när man gör en prototyp:



Figur 9. Olika synsätt vid prototyping.

Beroende på vilken målsättning man har kan prototyper skäras på olika sätt (*se figur ovan*). Ett snitt kan läggas på djupet (*vertikal*), t. ex. från databas till användarens gränssnitt, så att funktionerna inom en del av tillämpningen speglas, alternativt kan ett enklare *scenario* göras för exempelvis en arbetsuppgift. Ett snitt kan även läggas på tvären (*horisontell*), på så vis görs en prototyp på t. ex. användargränssnittet.

Vidare kan protypen syfta till att testa olika aspekter av systemet:



Houde S., Hill C., 1997 – What do Prototypes Prototype?

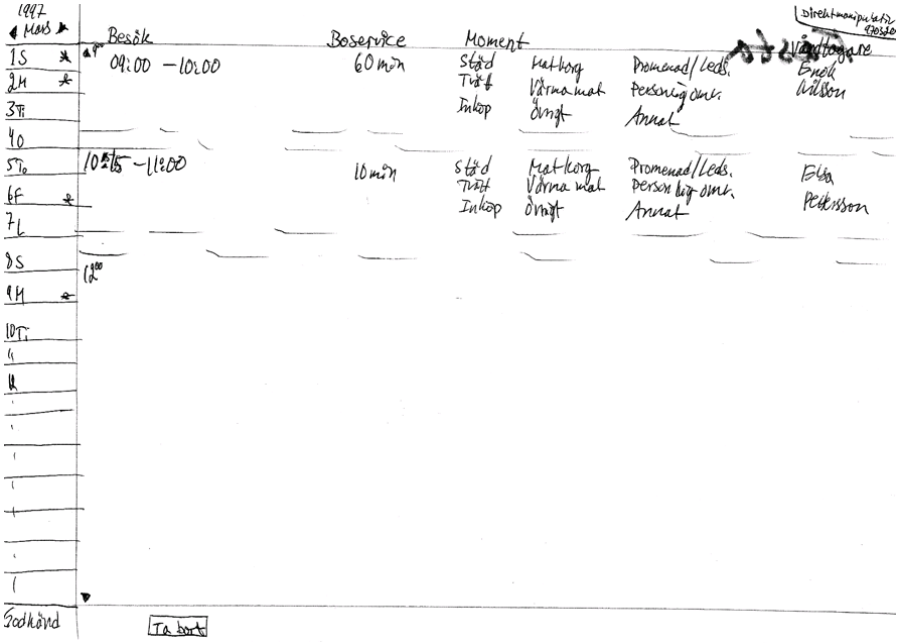
Figur 10. Prototyper av olika aspekter.

Här (*se figur ovan*) kan man se att de olika prototyperna (cirklarna) provar funktionalitet, implementation eller beteende/utseende. Dessa delar kan även mixas i prototypen; cirklarna ligger relativt mellan de olika hörnen, exempelvis lite funktionalitet och mycket beteende/utseende.

3.5.4 Exempel på arbetsprocess vid prototyping

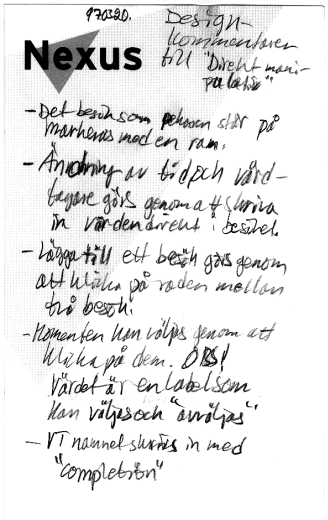
Prototyparbetet kan egentligen påbörjas när som helst. Beroende på vilket syfte som arbetet har (se ovanstående redogörelse) kan prototyper i olika skepnader introduceras tidigt eller sent. Vi kommer här att ge ett exempel på hur det kan gå till inom ett typiskt systemutvecklingsprojekt.

Efter inledande analyser av; syftet med systemet (affärsmål, processer m.m.), användare, arbetsuppgifter etc. påbörjas arbetet med att ta fram en konceptuell design av användargränssnittet. Detta sker t. ex. i form av designverkstäder (workshops) där användare deltar tillsammans med systemutvecklare och eventuellt andra intressenter. Här finns en rad olika metoder att tillgripa beroende på förutsättningar, resurstillgång etc. Syftet i detta skede är att pröva olika lösningar och relativt förutsättningslöst angripa problemen. Under den konceptuella designen tar man fram artefacts i form av exempelvis lösa pappersbitar som representerar olika delar av användargränssnittet. Dessa kan mycket enkelt skapas, sättas ihop och till och med slängas utan att man alltför tidigt binder eller låser sig för vissa lösningar. Man kan med fördel använda sig av s.k. blädderblock och notisar. Dessa artefacts blir stommen till de första mock-up:erna av användargränssnittet. Arbetet kan sedan fortskrida genom att man successivt förfinar sin design via mock-up:erna, skisser, detaljskisser och, när man tycker man kommit till en viss förutbestämd nivå, börjar flytta designen in i en datormiljö.



Figur 11. Exempel på tidig pappersskiss.

Man kan komplettera skisserna med enkla beskrivningar av hur användningen är tänkt. En teknik är s.k. storyboards, där skisserna följs av en beskrivning av arbetsflödet, bild för bild.



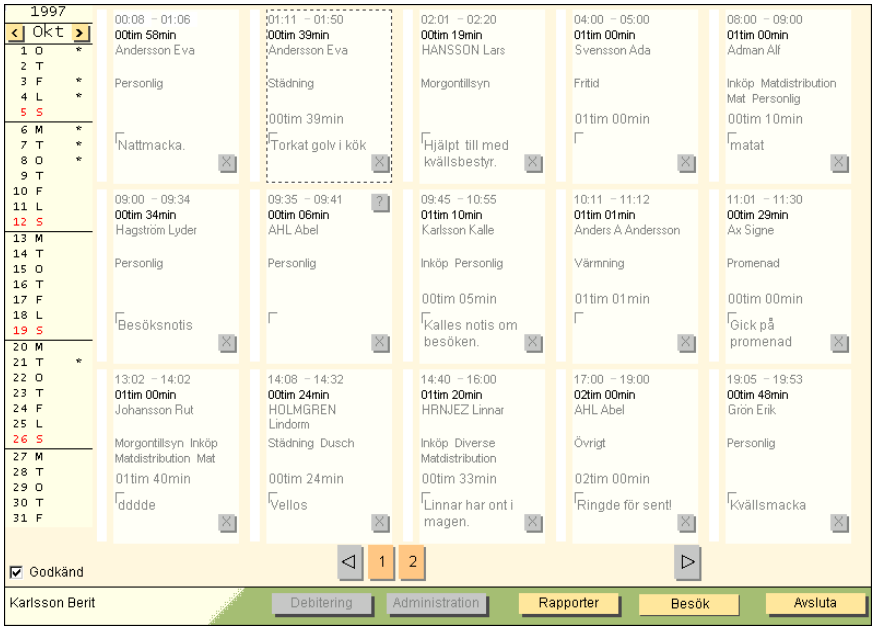
Figur 12. Enkel designkommentar.

Som ett stöd för prototyparbetet och designarbetet förordar vi att man efter hand dokumenterar de överordnade designbeslut som utvecklingsteamet kommer fram till. Detta blir ett viktigt redskap när man går vidare och blir mer och mer detaljerad. Istället för att behöva diskutera saker man redan bestämt så kan man hänvisa till en s.k. kriterielista och tagna designbeslut [Borälv & Göransson, 1997], exempel från ett projekt:

CRITERIA FOR PROTOTYPE #3:	DESIGN DECISIONS:
1. Fast scanning of content. 2. Easy and direct access. 3. Overview. 4. Support the average user.	• No navigation through menus. *Supporting criterion 2*. • Fixed and static layout. *Supporting criteria 1, 2 & 3*. • Clean layout without unnecessary graphical effects. *Supporting criterion 1*. • Focus on content. *Supporting criteria 2 & 4*. • Restricted use of metaphors, use only when appropriate. *Supporting criterion 4*.

Själva designprocessen kan dokumenteras på en rad olika sätt. QOC (question, option and criteria) [MacLean, 1991] är en variant, där olika problem (questions) förses med tänkbara lösningar (options) och designbesluten baseras på mer eller mindre vedertagna kriterier (criteria). Se gärna kapitlet *3.6 Användning av video och mock-ups* vad gäller andra förslag och ideer för dokumentation av en prototyps olika stadier.

I vårt lilla exempel blev den slutgiltiga designen följande:



Efter mock-up:erna och skisserna förde vi över designen i datormiljö med hjälp av det verktyg som var valt som utvecklingsverktyg för hela projektet. Arbetet bedrevs hela tiden i iterationer där lösningar provades av användarna.

En viktig aspekt att tänka på vid prototypingarbetet är att inte var rädd för att prova flera parallella prototyper. I det exempel vi redogjort för startade vi med tre prototyper som så småningom, via utvärderingar, »smälte samman« till ett användargränssnitt.

3.6 Användning av video och mock-ups

Under de senaste åren har användningen av videoteknik kraftigt ökat i system- och IT-utveckling. Orsaken till detta är förmodligen att tekniken på ett helt annat sätt är tillgänglig idag än för bara några år sedan, till ett rimligt pris. Videokamera är en naturlig del av ett användbarhetslabb men har idag fått en mycket bredare användning. Dessutom har det blivit betydligt enklare och billigare att redigera videofilm interaktivt i datorn.

VIDEO FÖR ANALYS. Att använda videokamera för att analysera en arbetsuppgift är ett kraftfullt hjälpmedel. Det är utmärkt för att dokumentera hur användarna idag agerar i sitt arbete. Genom att filma en användare som berättar vad man gör i sin arbetssituation samtidigt som användaren utför sitt arbete, ger ofta en helt annan bild av en arbetssituation en att bjuda in en användare och låta denna berätta vad man gör. Det har visat sig att låta ett antal utvecklare och/eller användare analysera en videofilm ger snabbt mycket mer information om en arbetssituation än att analysera intervjua/observera/modellera.

VIDEO FÖR DESIGN. Video är också ett brukbart hjälpmedel för att visualisera framtida designlösningar. Tack vare de enkla möjligheterna att visa prototyper så kan man snabbt skapa filmer som beskriver framtida scenarion med användande av ny teknik på ett mycket tidigt stadium, lång innan tekniken skapats.

VIDEO FÖR UTVÄRDERING. Att visa systemutvecklare hur användare i själva verket agerar med en prototyp är ett av de allra mest kraftfulla metoderna för att motivera systemutvecklaren att korrigera sina lösningar. Det finns många klassiska exempel där systemutvecklare »skriker och sliter sitt hår« när de ser hur »dumma« användarna är. Givetvis är inte användarna dumma men de fel som användare faktiskt gör vid interaktion med teknik är klart oförutsägbara.

VIDEO SOM DOKUMENTATION. Systemutvecklingsprojekt tenderar att producera enorma mängder dokumentation, och denna dokumentation är nästan bara användbar för de som själv har skrivit den. Det är vanligt att denna dokumentation inte i sig används av utvecklarna utan att de centrala punkterna

kommuniceras muntligt. Då är en videoinspelning en betydligt mer lättillgänglig dokumentation över en uppgiftsanalys, en prototyp eller en utvärderingssession.

3.7 Utvärderingsmetoder – med och utan användare

Användbarhetsutvärderingar kan utföras på en rad olika sätt, och med olika metoder. Här är en övergripande beskrivning av några de vanligaste – *expertutvärdering*, *fältstudie* (eller observationsintervjuer), *scenariobaserad* utvärdering, *gruppgranskning* och *laboratorieutvärdering*.

Expertutvärderingen kännetecknas av att den utförs av experter på användbarhet med erfarenhet av design och utveckling av användargränssnitt. Utvärdering utförs inte med direkt användarsamverkan, även om experten i förväg måste känna till användarnas arbetssituation, bakgrund, kunskaper etc. Vid utvärderingen granskar en expert systemet, program, manualer etc och provkör det om så är möjligt. Systemet utvärderas mot användargränssnittsriktlinjer s.k. Style Guides, checklistor och regler för användbarhet. Detta arbetssätt kännetecknas av att man snabbt och billigt upptäcker grundläggande användbarhetsproblem.

Fältstudier, eller *observationsintervjuer* utförs på system som redan är satta i drift. Arbetet utförs ofta inför en större systemrevision eller nyutveckling. Denna typ av utvärdering är den som är mest förankrad i verkligheten och ger en reell möjlighet att utvärdera ett system i en realistisk användarmiljö. Ett antal representanter från olika användargrupper väljs ut. Dessa får svara på ett några bakgrundsfrågor för att skapa användarprofiler. Användarna får dessutom svara på en enkät om hur de anser att systemet fungerar. Detta data kan sedan sammanställas och ge en ungefärlig bild av användaracceptansen. Fältstudien bedrivs hos användaren i dennes dagliga arbetsmiljö. Utvärderaren tillbringar tid hos användarna och sitter tillsammans med dem när de utför sitt arbete – observerar och frågar.

Scenariobaserad utvärdering utförs på system som håller på att utvecklas. Användarna får arbetsuppgifter, i form av scenarier, att utföra med hjälp av systemet. Denna typ passar utmärkt att använda i en iterativ systemutvecklingsprocess baserad på prototyping. Användarna studeras när de utför scenarierna och de får även svara på ett antal frågor om hur de upplever systemet.

En **gruppgranskning** utförs med representanter från användarna och utvecklarna, men även ergonomer, dokumentatörer etc. kan delta. En expert på användbarhet leder arbetet. Gruppen går igenom ett antal scenarier som bygger på användarnas arbetsuppgifter. Man diskuterar varje steg som användaren

måste utföra. Funna problem eller tveksamheter noteras, kategoriseras och värderas. Gruppgranskningar genomförs under en begränsad och kort tid. Normalt inte mer än ett par timmar. Används tidigt i ett utvecklingsprojekt. Fördelarna med gruppgranskning är att man får många infallsvinklar på systemet. Viktigt att tänka på är att alla i gruppen får komma till tals och värderas förutsättningslöst. Man bör se till att arbetet bedrivs konstruktivt, och inte leder till att de olika aktörerna enbart ägnar sig åt bevaka sina egna intressen!

Laboratorieutvärdering genomföras i s.k. »usability labororatories«. Vid laboratoriekörningen är utvärderarna passiva åskådare som »bara« samlar in data. Data analyseras först efteråt. Arbetet går till så att användaren får ett antal uppgifter att lösa. Utvärderaren studerar hur uppgiften löses, ser vilka problem användaren ställs inför etc. Det är viktigt att inga utvecklare är med vid detta tillfälle eftersom de har en benägenhet att försöka hjälpa användarna till att använda systemet på »rätt« sätt och inte ett »naturligt« uppgiftsrelaterat sätt. I laboratoriet finns utrustning typ video för att fånga allt som användaren gör.

Sammanfattningsvis kan man säga att de olika typerna av utvärdering kan genomföras enskilda eller i kombination. Detta beroende på ambitionsnivå och målsättning. Generellt – en *expertutvärdering* är den som oftast går snabbast att genomföra men som å andra sidan inte ger full relevans till användarens uppgiftslösning. *Laboratorieutvärderingen* är vanlig vid utveckling av nya system, men förekommer även för system i drift. Bristen med denna typ av utvärdering är att den utförs i en annan miljö än vad användarna normalt är vana vid samt att den är relativt kostsam, kräver speciella lokaler och utrustning. *Fältstudien* ger oftast det »bästa« och mest relevanta resultatet, men kan vara lite kostsam då man går in och »stör« användarna i deras dagliga arbete. En annan nackdel kan vara att den bara går att utföra på system som är i någon form av drift. Den *scenariobaserade* passar bäst vid nyutveckling och prototyping, men kan även användas vid revisioner etc. *Gruppgranskningar* är relativt effektiva om de genomförs på ett konstruktivt sätt. Gruppens sammansättning och deltagarnas inställning är oerhört viktiga parametrar för resultatet.

4

Modeller för systemutveckling

Systemutveckling kan bedrivas med olika »företeelser« i centrum. Ganska ofta sätter man en viss teknik i centrum; det är en viss teknisk lösning som skall föras fram. I andra lägen är det data i sig som är i centrum; exempelvis en databas. Ett annat sätt att se på saken är att säga att det viktigaste är att de som *skall* använda systemet också *kan* använda systemet. För att nå dit är det viktigt att sätta *användarna* i centrum. I och med detta säger man också att det är den verksamhet som användarna bedriver som man skall fokusera på. I förlängningen eftersträvar man största möjliga *verksamhetsnytta*.

> *– Eighty percent of software life cycle costs occur after the product is released, in the maintenance phase. Of that work, 80% is due to unmet or unseen user requirements only 20% of this is due to bugs or reliability problems.*
>
> Karat, C. (1993), Usability Engineering in Dollars and Cents, IEEE Software, May 1993, pp 89.

– Without user-centred design, a user interface typically has around forty flaws that can slow users and lead to error.

Landauer, T.K. (1995), The trouble with Computers, Cambridge, Mass: MIT Press, pp 223.

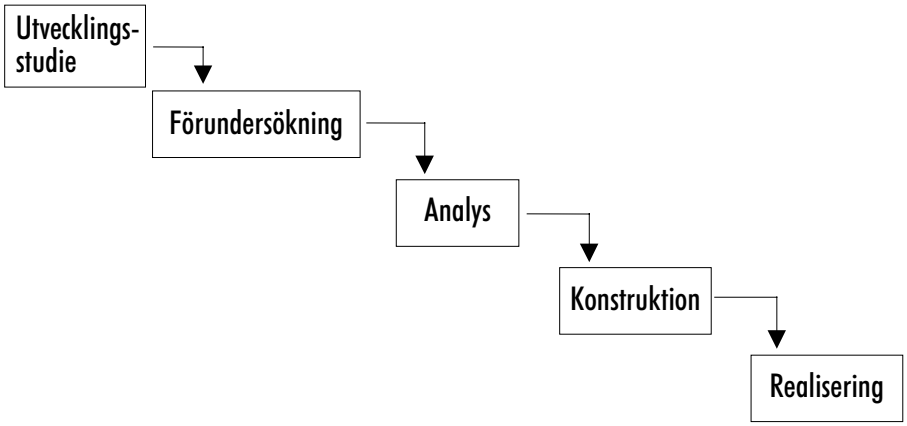
– Norwich Union, an insurance company in Australia, found that calls to its help desk reduced dramatically by two thirds after one of its core applications was improved using user-centred design techniques.

Norwich Rethinks Customer Service, Computer World, 24 November 1995.

4.1 Systemutvecklingsmodeller – vattenfall kontra iterativa

Traditionell systemutveckling bygger ofta på den s k vattenfallsmodellen [Boehm, 1976]. Bland de klassiska modellerna som använder sig av vattenfallsmodellen kan nämnas SAK – strukturerad analys och konstruktion av informationsbehandlingssystem [Wigander, Svensson, Schoug, Rydin & Dahlgren, 1979]. SAK-metoden, i likhet med andra modeller som är baserade på vattenfallsmodellen, bygger på principen att systemutveckling är en ingenjörsverksamhet och bör läggas ut som en sekvens av aktiviteter vilka kan planeras in i tiden och som resurser kan allokeras efter. En sådan modell kan synas vara idealisk för att lösa ett problem, men verkligheten ter sig oftast annorlunda.

Vattenfallsmodellen innebär att ett systems utvecklingscykel är uppdelad i faser som bildar en sekvens. Ett grundantagande är att ett problem kan struktureras på ett tekniskt och logiskt sätt tidigt i processen, och sedan lösas steg för steg.



Figur 13. Exempel på vattenfallsmodellen för systemutveckling

Denna typ av traditionell systemutvecklingsmetodik är fortfarande mycket vanlig men passar inte särskilt bra vid användarcentrerad utveckling. Anledningarna till detta är många och man kan bl a peka på följande [Budde, Kautz, Kuhlenkamp, Züllighoven, 1992]; systemutveckling ses till största delen som ett tekniskt problem, aktiviteterna i projektet är arrangerade i en strikt sekvens, alla krav på systemet skall vara klara innan någon utveckling börjar etc. Detta återspeglar ett synsätt där man utgår ifrån att alla problem går att formulera och strukturera på ett logiskt sätt (helst strikt matematiskt) samt att resultatet är mer eller mindre förutsägbart. Bland de svårigheter eller problem som detta synsätt inte tar hänsyn till kan nämnas [Budde et al, 1992]:

- En fullständig, komplett och statisk kravspecifikation går inte att skapa innan utvecklingen börjar.
- Specifikation, design och konstruktion av ett system kan inte ses som separata arbetssteg över tiden.
- Stora och komplexa formella specifikationer är för det mesta obegripliga, både för användare och utvecklare.
- Användarna stängs ute från utvecklingsprocessen.

Rational Unified Process – RUP [Kruchten, 1998] beskriver vattenfallsmodellen och dess problem enligt följande:

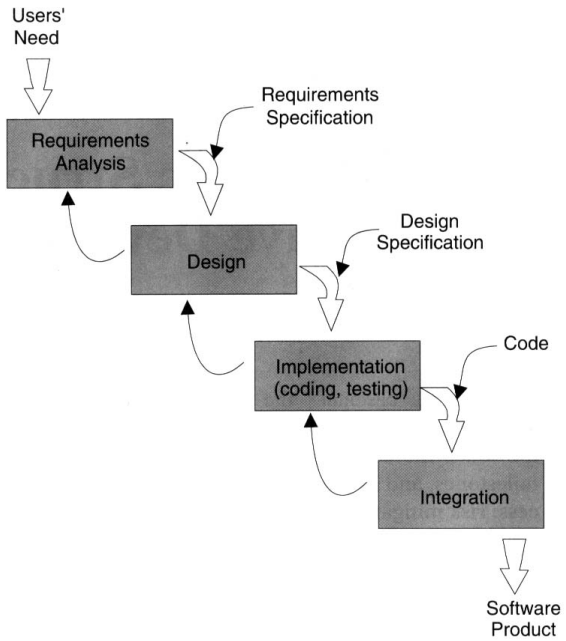


FIGURE 4-1 *The sequential process*

Figur 14. Vattenfallsmodellen enligt RUP.

Man poängterar att det finns två stora feluppfattningar vad gäller synen på systemutvecklingsprocessen:

- Många jobbar med s k frysta kravspecifikationer. Detta leder till stora problem eftersom:
  - Användarnas behov förändras
  - Problemen förändras
  - Tekniken förändras
  - Marknaden förändras
  - Vi kan inte uppnå tillräcklig detaljeringsgrad
- Många tror att det går att beskriva allting på papper innan utvecklingen. Detta visar sig vara nästan omöjligt i realiteten.

Inom RUP tar man istället ställning för att bedriva utvecklingen i form iterationer, där varje iteration i sig är ett mindre och mer hanterligt vattenfall. Mer om detta i kapitlet *7.1 Rational Unified Process*.

Utvecklandet av användbara system kräver en helt annan modell (än vattenfallsmodellen) för systemutveckling där användarna står i centrum. För att ha förutsättning att utveckla system med god användbarhet formulerade pionjärerna Gould och Lewis ett antal grundprinciper [Gould & Lewis, 1983; Gould & Lewis, 1985]. Dessa principer sammanställdes av Poltrock och Grudin [Poltrock & Grudin, 1994; Katzeff & Svärd 1995] för att ingå i en studie kring användbarhetsmognaden hos systemutvecklingsföretag:

Tidigt fokus på användarna och deras uppgifter. Designern måste förstå vilka användarna kommer att vara, deras kognitiva, beteende- och attitydmässiga inställning, samt arbetets karakteristik.

Interaktiv design. Typiska användare måste bli en del av utvecklingsteamet från första början.

Empirisk mätning. Experiment med prototyper med vilka riktiga användare gör riktiga arbetsuppgifter i syfte att deras reaktioner och attityder skall observeras, samlas in och analyseras.

Iterativ design. En cyklisk process av design, utvärdering och ny design skall upprepas så ofta som det är möjligt.

Den grundläggande tanken är att samtliga principer skall vara uppfyllda för att ett utvecklingsprojekt skall kunna betraktas som användarcentrerat. Detta är givetvis en ganska idealiserad bild över hur man skulle vilja bedriva utveckling. Problem uppstår när detta synsätt skall föras samman med idag vanliga och mer

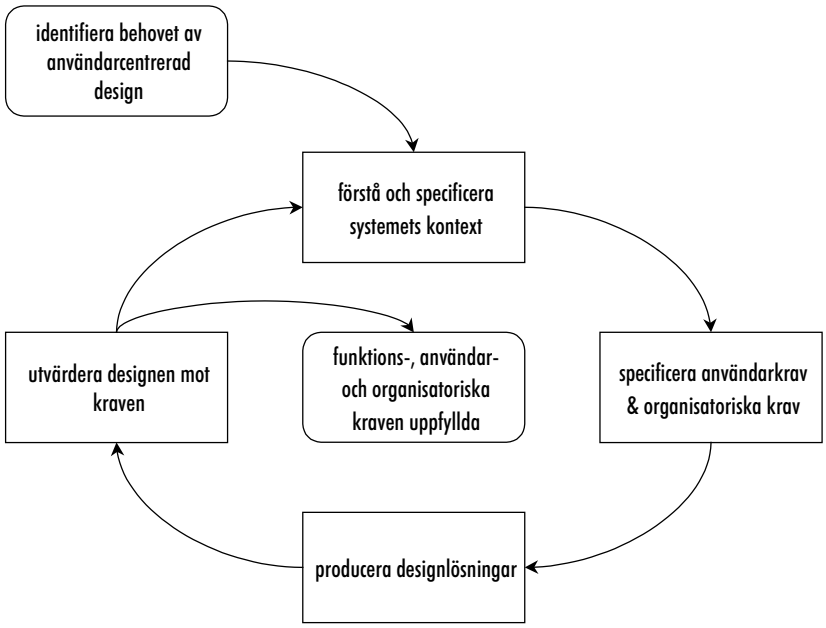
eller mindre vedertagna arbetssätt som t ex tidiga kravspecifikationer. Detta är vanligt i näringslivet eftersom kravspecifikationerna ofta är det affärsbärande dokumentet. Att arbeta med »frysta kravspecifikationer« motsäger ett användarcentrerat och iterativt arbetssätt eftersom man i förväg inte känner alla krav i detalj och inte heller hur problemen skall lösas. Ett iterativt arbetssätt ställer också stora krav på kontinuitet och dynamik inom projektet, både med avseende på personal och aktiviteter – se t ex »spiral model« [Boehm, 1988].

Gould och Lewis synsätt har nu anammats som grunden i en internationell standard (ISO 13407 – *Human-centred design processes for interactive systems*). Den använder sig av följande fyra punkter som grund för definitionen av användarcentrerad design:

a) Aktivt deltagande av användare och en tydlig förståelse av användarnas uppgiftskrav.
b) Lämplig fördelning av funktioner mellan användare och teknologi.
c) Iterationer av designlösningar.
d) »Mångvetenskaplig« design.

Vidare definierar man fyra användarcentrerade designaktiviteter som skall utföras inom ett systemutvecklingsprojekt:

1. Förstå och specificera systemets kontext.
2. Specificera användar- och organisatoriska krav.
3. Producera designlösningar.
4. Utvärdera om designen uppfyller kraven.



Figur 15. ISO 13407 – Human-centred design processes for interactive systems.

Initialt måste man ta ställning till om det förestående utvecklingsprojektet skall bedrivas efter en användarcentrerad modell eller efter någon annan typ av utvecklingsmodell. När väl detta är bestämt går man vidare och tittar på vilka användargrupper som finns, vilka arbetsmål användarna har och i vilken miljö som användarna skall använda systemet. Vid kravsammanställningen tittar man sedan inte bara på funktionella krav utan lägger lika stor vikt vid kraven på användbarhet. Dessa användbarhetskrav skall uttryckas, så långt det är möjligt, i termer av mätbara mål. De designlösningar som tas fram skall vara konkreta och möjliga att förstå för amvändarna (simuleringar, mock-ups etc). Lösningarna skall även provas av användare i så realistiska arbetsuppgifter som möjligt. Designprocessen skall sedan itereras tills dess att en förutbestämd nivå av kraven har uppfyllts. Vid slutet av varje iteration skall man sedan genomföra mer formella utvärderingar mot de användbarhetsmål man satt upp i tidigare steg. Dessa utvärderingar ger ovärderlig återkoppling in i nästa iteration. Itererandet kan avbrytas när man uppnått de förutbestämda målen.

Det är viktigt att påpeka att ISO 13407 inte är en färdig systemutvecklingsmodell utan representerar ett koncept som kan omsättas i processer och metoder samt inordnas i andra, redan befintliga, systemutvecklingsmodeller.

Den iterativa processen är en stor del av nyckeln till en lyckad systemutveckling och en grundbult i ISO 13407. Ett iterativt arbetssättet:

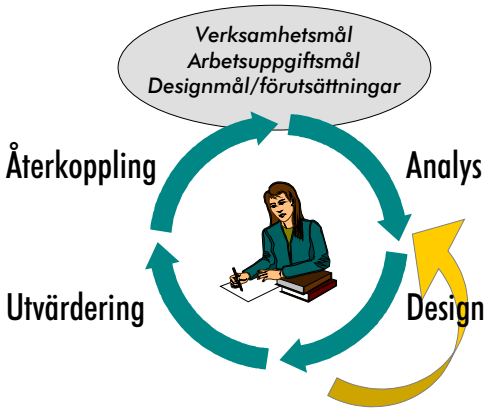
- Tar hänsyn till användarmedverkan

- Bygger på upprepade användartester
- Bygger starkt på prototyping
- Tar vara på den kreativa processen att skapa något
- Fokuserar på användarna och deras arbetsuppgifter
- Använder användarorienterade representationer

Som visats bygger ett användarcentrerat arbetssätt på ett iterativt arbetssätt. Det är dock inte helt enkelt att bedriva sådan utveckling och hantera projekt enligt principen om stegvis förfining av systemet. Några viktiga aspekter kring detta tas upp senare i rapporten.

4.2 Systemskiss och iterativ utveckling

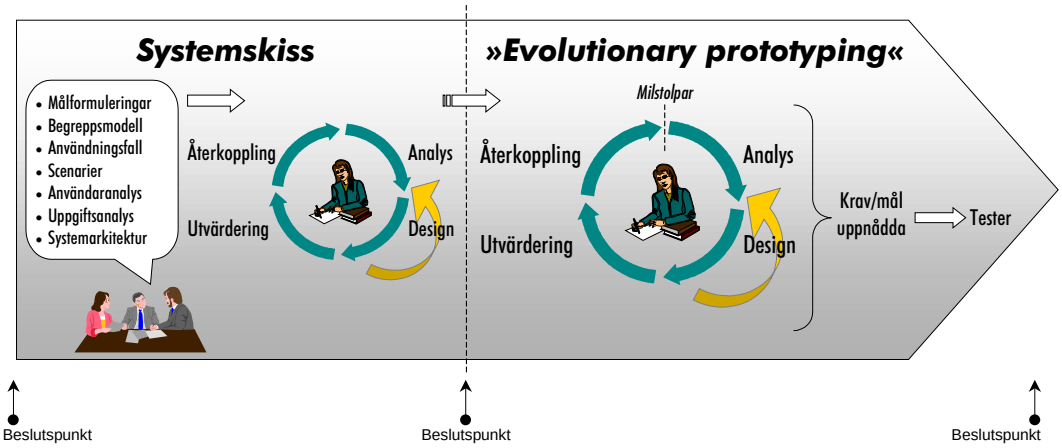
Om vi nu kan konstatera att vattenfallsmodellen inte ger oss särskilt bra förutsättningar för att utveckla användbara system, utan vi bör välja ett iterativt och användarcentrerat angreppssätt, hur tar vi oss då dit? Till att börja med kan vi titta på hur man kan utgå ifrån ISOs standard och försöka omsätta den i en mer konkreta systemutvecklingsmodell. ISOs standard representerar ju ett koncept som i sig inte ger en färdig utvecklingsmodell. Nedan finns en skiss som är en schematisk beskrivning av en iterativ och användarcentrerad systemutvecklingscykel.



Figur 16. Principiell skiss över en användarcentrerad utvecklingscykel.

Utifrån en avgränsning vad systemet skall stödja i termer av processer, delprocesser m.m., sätter man upp målsättningar och förutsättningar för systemet. Arbetet bedrivs sedan i två iterativa cykler, där den »större« cykeln genererar inkrement av systemet och den »mindre« itererar tänkbara designlösningar. Detta är naturligtvis en oerhört förenklad bild, men ger ändå en uppfattning om hur arbetet skulle kunna gå till.

Om vi vidareutvecklar denna bild och sätter in den i ett större sammanhang kan det se ut så här:



Figur 17. Exempel på modell för att jobba iterativt och användarcentrerat. Observera att systemskissdelen bör motsvarar cirka 10%–15% av helheten.

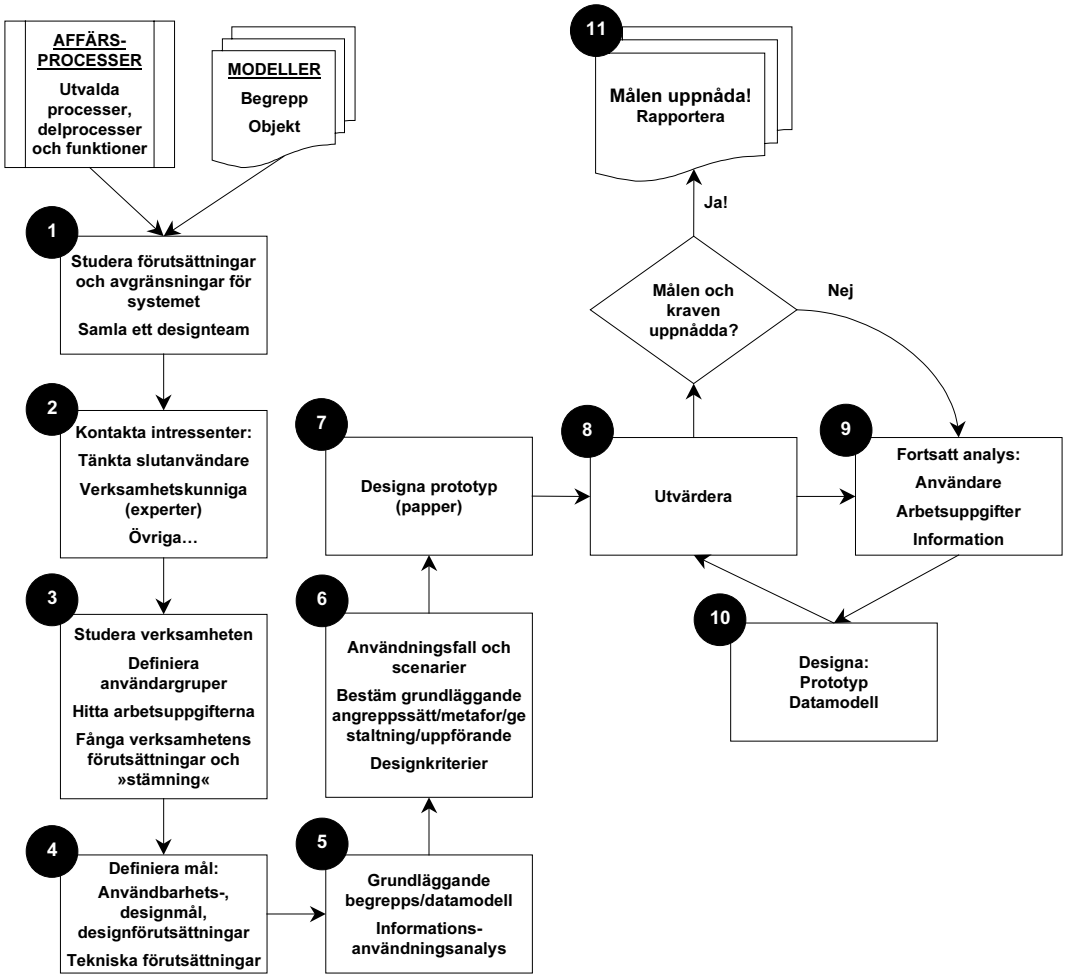
Här har vi delat upp utvecklingsarbetet i två huvuddelar: systemskiss och evolutionary prototyping. Systemskissen är tänkt som målformulerings och kravinsamlingsfas där arbetet bedrivs relativt förutsättningslöst i nära samarbete med användarna. Här fokuserar man på att söka sig fram till vad systemet skall stödja för typ av arbetsuppgifter och vilka egenskaper detta system skall ha. Arbetet bygger mycket på tidig prototyping och korta iterationer.

Systemskissen leder fram till ett förslag på systemlösning, vilken sedan kan fortsätta att utvecklas och realiseras i form av evolutionary prototyping. En mycket viktig aspekt att tänka på är att inte falla i in problemen som finns i vattenfallsmodellen vad gäller att få in allting på en gång. Med ett iterativt arbetssätt har man möjlighet och bör eftersträva att styra projektet med hjälp av funktionalitet och inte hela tiden flytta tidsramarna, s.k. »time-boxing«. Detta betyder att man alltid försöker leverera i tid och istället låter eventuell ny, eller försenad funktionalitet komma i nästa iteration.

Proportionerna i bilden ovan vad gäller tiden inom respektive fas är inte korrekta. Detta beroende på att vi vill få in båda faserna i en bild, och ändå få plats med detaljerna. En uppskattning är att systemskissfasen i själva verket är mellan 10%–15% av helheten.

4.2.1 En nedbruten arbetsprocess för en systemskiss

Vi kan försöka att gå ytterligare lite djupare in i systemskissen och se vilka moment och aktiviteter den skulle kunna innehålla:



Figur 18. Ytterligare en variant på systemskiss.

Figuren kan ge sken av att processen är strikt sekventiell, men så är naturligtvis inte fallet. Som vi har försökt beskriva genom hela rapporten så skall arbetet bedrivas iterativt och med stor användarmedverkan. Dock kan det i vissa sammanhang krävas att man »lägger ut processen« mer som en sekvens av händelser för att kunna identifiera de aktiviteter som skall utföras.

Vi vill också poängtera att arbetsprocessen *inte* tar upp någonting om *projektstyrningen* eller exakt hur resultatet levereras, resurser allokeras etc. Fokus ligger på ett antal konkreta arbetssteg som bör ingå vid framtagandet av t. ex. systemskiss eller en prototyp.

1. **Utgå från eventuella processbeskrivningar och begrepps-/objektmodeller**

   Studera förutsättningarna för systemet. Se vilka avgränsningar som finns och sätt er in i vilka delar av verksamheten som systemet skall stödja. Börja samla det team som skall ta sig an uppgiften att utveckla systemet. Tänk på de olika kompetenser och specialister; användare, utvecklare m fl som skall ingå.

2. **Kontakta målverksamheten**

   Förbered målverksamheten på att ni kommer att starta ert arbete och att ni kommer att »störa« dem med intervjuer etc. Det är oerhört viktigt att förankra arbetet i verksamheten och se att alla jobbar »åt samma håll«.

3. **Studera målverksamheten**

   Gå ut till målverksamheten och studera användarna. En förutsättning för att kunna göra ett användbart system är att man går ut i verksamheten och analyserar det arbete som fortgår där. Detta inkluderar de användaranalyser och uppgiftsanalyser vi tidigare tagit upp.

4. **Definiera mål**

   Försök att definiera den användbarhetsmål som skall gälla. Gå även vidare och se vilka övergripande designmål som skall finnas. Se också till de olika förutsättningar som finns för t ex designen och tillgänglig/hänvisad teknik.

5. **Begrepps- och datamodell**

   Se till att de informationsmängder som användarna har behov av initialt läggs in i en begreppsmodell (konceptuell modell). Denna är ett bra underlag för se vilken information som skall vara tillgänglig i exempelvis användargränssnittet. Den ger oss också möjlighet att se till att alla inom projektet, i stort, menar samma sak med verksamhetens olika begrepp. Vi kan dock inte stanna vid detta utan måste också gå vidare med informationsanvändningsanalys där vi kan se mer detaljerat när, hur etc användarna har behov av information.

6. **Användningsfall och scenarier**

   Här börjar vi formulera mycket övergripande hur systemet skall stödja användarna i deras arbete. Det kan ske i termer av användningsfall, scenarier, storyboards m m. Vi börjar också fundera på övergripande designkriterier: skall vi använda någon metafor; vilka huvuddelar skall användargränssnittet innehålla; vad skall användaren se etc.

7. Designa prototyp

I samband med ovanstående punkt 6, börjar vi skissa på hur användargränssnittet kan se ut. Vi jobbar tillsammans med användarna i t ex designverkstäder.

Punkt 8, 9 och 10 är sedan en iterativ cykel av: analys–design–utvärdering. Här itererar vi till dess att våra mål är uppnådda. Exakt antal iterationer varierar med de förutsättningar som projektet har. Här kan tid, resurser etc styra. Det viktigaste är att man i förväg har bestämt sig för med vilka kriterier man skall avbryta det iterativa arbetet.

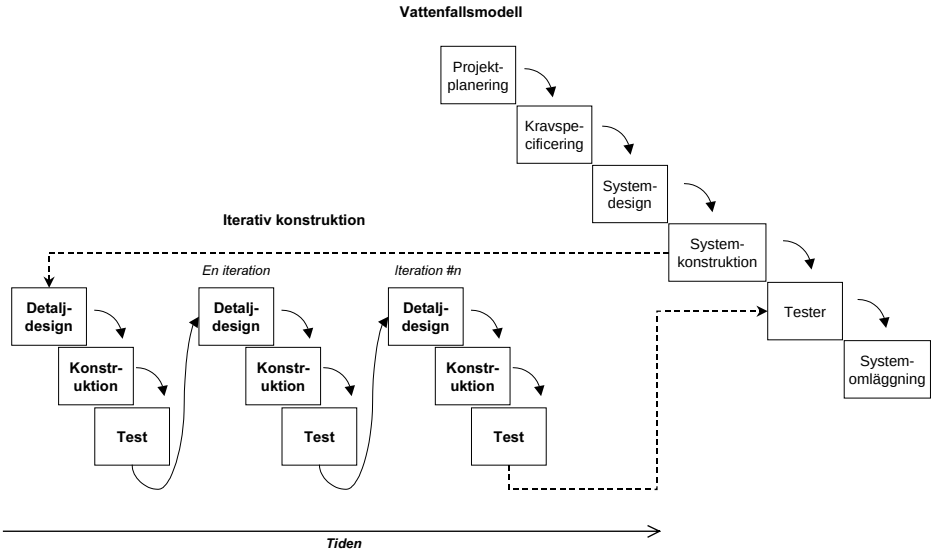
11. Målen uppnådda, rapportera

Beroende på projektdirektiven är det nu dags att rapportera resultatet av vårt arbete. Här kan det ligga en beslutspunkt om att gå vidare, eller också tycker man att det räcker med den systemskiss som nu tagits fram.

Ovanstående process och dess aktiviteter kan tjäna som utgångspunkt för arbetet med en systemskiss, men kan naturligtvis anpassas ytterligare efter de förutsättningar som finns för varje enskilt projekt.

4.2.2 Iterativ konstruktion i en vattenfallsmodell

Även om man olika anledningar följer, eller är hänvisad att följa, en vattenfallsmodell finns vissa möjligheter att kunna leverera systemet i delar. Här är en variant med en iterativ konstruktionsfas, i en vattenfallsmodell.



Figur 19. Iterativ konstruktion i en vattenfallsmodell.

Det är naturligtvis ingenting vi förordar, men kan fungera i en övergångsfas när en organisation exempelvis går ifrån ett vattenfall till en iterativ utvecklingsprocess.

Eftersom det är stora problem med att få system klara i tid inom resursramar etc. är det viktigt att försöka dela upp konstruktionsfasen i delar (inkrement) som kan testas och provas innan hela systemet är färdigt. Det är även eftersträvansvärt att kunna prova dessa inkrement i målverksamheten. Detta är dock inte alltid möjligt, men bör noga övervägas.

4.3 Kravspecifikation – ett affärsbärande dokument

Kravspecifikationen har en speciellt roll inom systemutvecklingsprojekt. Dels innehåller den naturligtvis kraven på systemet, men den har också en roll som ett affärsbärande dokument. Detta innebär bl. a. att kravspecifikationen blir ett mycket formellt dokument som i värsta fall skall »hålla« vid en rättstvist. Detta är ingen bra situation – att blanda ihop affärer/juridik med förutsättningarna för att skapa ett användbart system. Vidare, kravspecifikationerna är dessutom oftast uttryckta i funktionella krav och täcker inte in icke funktionella krav. Icke funktionella krav är mycket viktiga och kan t. ex. betyda saker såsom en användares mål med sin arbetsuppgift.

Förslag på lösningar kan vara att:

- Jobba mer utifrån målformuleringar och istället låta kravställandet bli en effekt av att man försöker uppnå målen med designlösningar.
- Jobba med prioriterade kravlistor uttryckta i exempelvis termer av vad som *måste*, *bör*, *kan* och *är bra* om det ingår i systemet. Dessa listor tas fram för varje utvecklingsiteration och skiljs från affärsstyrande dokument.

4.4 Verksamhetsutveckling kontra systemutveckling

Problemet när man bestämmer sig för att utveckla ett nytt datorstöd är ofta ett organisatoriskt problem. Ett beslut fattas om att utveckla ett nytt (eller en ny version av ett) datorstöd för en viss verksamhet eller arbetsuppgift. Ett sådant beslut kan vara grundat på ett behov av att uppdatera tekniken (befintligt system är så gammalt att man helt enkelt inte kan uppdatera det/ kompetensen i hur systemet är uppbyggt eller i verktyget som systemet är byggt finns inte längre i organisationen), eller som ett resultat av visioner om framtiden som tagits fram lokalt eller i en större del av verksamheten. Problemet är att dessa visioner sällan är framtagna med de nya möjligheter som dagens IT erbjuder i åtanke.

Vad menar vi med visioner och framtidsscenarier? Det finns idag inte tillräckliga kunskaper, eller förutsättningar i övrigt, för att i detalj precisera och utforma morgondagens system för en viss verksamhet, exempelvis hantering av skatteärenden. Som ett led i arbetet med att ta fram sådan kunskap är det viktigt att klargöra vilka förutsättningar som gäller för »morgondagens verksamhet« och för planeringen och styrningen av denna; vilka krav man kommer att ställa på skatteverksamheten; hur organisation, informations- och IT-system kan och bör utformas m.m. Det är detta vi menar med att formulera visioner och framtidsscenarier. En vision kan vara en idé eller en uppfattning om något enskilt förhållande i framtiden. Med ett scenario menar vi en mer sammansatt framtidsbild som beskriver viktiga aspekter på den framtida verksamheten i något helhetsperspektiv.

De formulerade scenarierna ska kunna utgöra en bas för kommande analys- och utvecklingsarbete. Kan vi beskriva nuläget samt en framtidsbild i samma termer, kan en jämförelse mellan dessa sägas utgöra en grund för vilken utveckling som behövs för att uppfylla visionerna. Scenarierna kan också användas när vi vill utvärdera prototyper av olika slag, t ex prototyper av möjliga presentationsformer i framtida användargränssnitt.

Vi kan särskilja två olika slags visioner, dels hur vi *tror* att framtiden kommer att se ut, dels hur vi *vill* att den blir. Bägge dessa aspekter är viktiga att fånga upp. Det vi *tror* är vår tolkning av den utveckling vi idag ser som trolig. Det vi *vill* är en konkretisering av den utveckling vi bedömer vara den för verksamheten som helhet mest önskvärda, för att uppnå »bra« lösningar i framtiden. Detta speglar våra förväntningar på den »bästa« utvecklingen. Det är viktigt att alla parter som har intressen i denna utveckling ges möjligheter att deltaga i utformningen av framtidsbilderna. De som bör deltaga i visionsarbetet är sådana personer inom verksamheten som förväntas kunna bidraga på ett konstruktivt och kreativt sätt till att formulera tänkbara framtidsscenarier.

Ambitionen skall inte vara att formulera kompletta och färdigbearbetade bilder, utan se detta som en del av en process som kommer att pågå framöver.

En grov struktur för vilka olika aspekter på verksamheten som kan studeras är:

- viktiga omvärldsfaktorer som påverkar förutsättningarna,
- ramar och begränsningar för utvecklingen,
- mål och förväntningar på morgondagens skatteverksamhet,
- viktiga problem som måste lösas i framtiden,
- hur förändras styrmålen?
- teknik för morgondagens IT-system,

- framtida arbetsorganisation, roller etc,
- kompetenser hos berörd personal, utbildning,
- informationsbehovet i olika arbetssituationer,
- tekniklösningarna, hur ser morgondagens tekniklösningar ut?
- presentationsformerna, design av användragränssnitt,
- arbetsmiljökrav m.m.

När ett beslut fattas om systemutveckling vidtar ett mycket långt analysarbete i syfte att utveckla verksamheten, snarare än att utveckla IT-stöd.

Projekten är alldeles för långa och stora. Ett exempel från RSV: Ett projekt startades hösten 1996 men i huvudsak på började arbetet hösten 1997. Ett nytt system för kronofogdens indrivningsverksamhet är utlovat till oktober 2001. Mer än halvvägs i projektet hade man ännu inte sett någon form av skärmbild eller ens utkast till skärmbild. De som ansvarar för gränssnittet i projektet sände våren 1999 tillbaka användningsfallen eftersom de var för »yviga« för att man skall kunna göra gränssnitt eller gränssnittsmodellering baserat på dem. Först under hösten 1999 gick man in en iterativ fas med analys–design–utvärdering i cykler om åtta veckor (eftersom detta var de cykeltider som RFV bestämt sig för i ett referensprojekt som man kollat upp). Då började man att analysera sina användningsfall. Det har hela tiden varit tillsagt att man skall undvika att rita skärmbilder för tidigt, för att man kan riskera att användaren låser fast sig vid ett utseende. Givetvis finns det poänger bakom sådana argument men man borde istället ha fokus att påbörja design så tidigt som möjligt och kanske rent av ha prototyping som en metod att ta fram kraven för systemet.

5

Aspekter på projektarbete

5.1 Vad är en utvecklingsmodell?

Nästan allt IT-utvecklingsarbete sker idag med hjälp och stöd av en modell för hur arbetet skall bedrivas. Det kan vara en systemutvecklingsmodell för systemutvecklingsarbetet, en projektstyrningsmodell för styrning av projektarbetet eller en förvaltningsmodell för förvaltningsarbetet. En sådan modell är viktig för att beskriva och lära ut det arbetsmönster som organisationen har. Det har under lång tid varit vanligt att varje större organisation som bedriver systemutvecklingsarbete etablerar en egen modell för detta arbete, Enator har sin PAS-modell, WM-data och Ericsson sin Delta-modell, RSV har sin Skruv-modell, osv. I det senaste har det varit vanligt förekommande att utvecklingsorganisationerna, erkännande de problem som bevisligen förekommer i utvecklingsarbetet, beslutat att revidera och oftast låta upphandla en ny utvecklingsmodell. Detta skapar en marknad där mångfalden utvecklingsmodeller minskar vilket givetvis innebär att kompetensen ökar, så

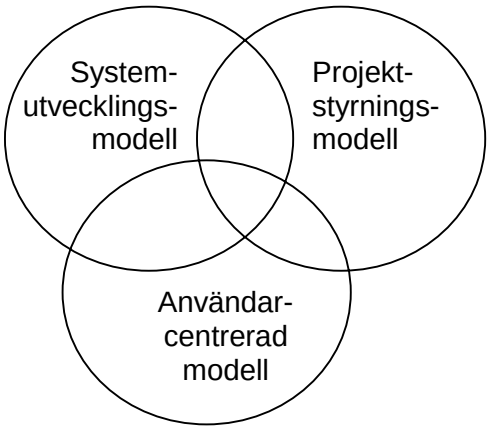
att man enklare t ex kan köpa in en konsult utan att denne behöver ges en upplärningstid.

Ett problem i detta är att även om det finns en modell så tenderar man inte följa en modell till punkt och pricka. Vilket arbetsmönster man använder i utvecklingsarbetet beror så mycket på de egna erfarenheter man har. Detta får till följd att variationen i arbetsmetoder blir mycket stor trots att man säger sig följa en och samma utvecklingsmodell. Det visar sig i de fall vi har bett människor som varit ansvariga för utvecklingsarbete att rekonstruera hur man arbetat att detta är mycket svårt. Man tenderar att ange modellen som den arbetsmetod man följt och de principer som praktiskt har utnyttjats är i princip tyst kunskap. Dessa observationer väcker frågor om vilket syfte en modell faktiskt skall ha, och vilken grad av detaljstyrning en modell skall innehålla.

Ett annat problem som vi tydligt sker i detta arbete som reviderar systemutvecklingsmodellen i organisationen är att man förväntar sig att den nya systemutvecklingsmodellen skall innehålla svaret på alla frågor. Detta är givetvis inte fallet. Den generella systemutvecklingsmodellen som fungerar i alla olika typer av projekt inom alla upptänkliga organisationer finns inte. Modellerna utger sig inte heller för att vara en allmängiltig modell utan meningen är att de skall anpassas till olika typer av projekt och organisationer.

I vårt studium av vilka utvecklingsmodeller som styr IT-utvecklingen i en organisation kan man i tillägg till vad som sagts i tidigare avsnitt konstatera att:

1. Det inte är någon tydlig skillnad mellan verksamhetsutveckling och systemutveckling. Verksamhetsutveckling bedrivs ofta med likartade modellerings- och utvecklingsmetoder som systemutvecklingen dock oftast utan inslag av teknikkompetens. Verksamhetsutvecklingen skapar inte de mål och visioner för den framtida verksamheten som de borde skapa utan alstrar snarare teknikutvecklingsprojekt som om detta hade varit ett underförstått syfte ända från början. När sedan ett systemutvecklingsprojekt startas måste man oftast inleda detta med verksamhetsutveckling för att förstå vilka mål som systemet skall stödja. Det är dock ofta inte ovanligt att systemutveckling bedrivs utan att man har en klar målsättning för arbetet.
2. Många av de aspekter som blir viktiga för användbarheten och användarcentreringen av utvecklingsarbetet ligger inte i systemutvecklingsmodellen enbart utan ofta i den projektstyrningsmodell som organisationen följer. Projektstyrningsmodellen är oftast en isolerad företeelse i jämförelse med systemutvecklingsmodellen.



Figur 20. Modellerna för projektstyrning och systemutveckling går i varandra.

Observationen är entydig; systemutvecklingsmodellen och projektstyrningsmodellen har starka beroenden. Hur stort snittet mellan dessa är och vilka aspekter som snittet innehåller beror till stor del på vilken systemutvecklingsmodell man väljer. T ex så anger DSDM (Dynamic Systems Development Method) relativt detaljerat vilka olika kompetenser som finns inblandade i projektet och hur ofta delleveranser skall göras. Vilket är något som ofta projektstyrningsmodellerna behandlar.

Komplexiteten för en användare/utvecklare att välja systemutvecklingsmodell ökas i och med att de på marknaden befintliga modellerna inte är jämförbara. Det finns olika orsaker till detta:

- Modellerna spänner över olika avsnitt av utvecklingsarbetet. Vissa modeller fokuserar på tidiga faser, andra har sin tyngdpunkt i andra faser.
- Vissa modeller är mer teknikstyrda, andra har mera ett användbarhetsfokus.
- Vissa modeller gör anspråk på att vara detaljerade metoder med metodsteg för alla delar i en viss sekvens, andra är mer ramverk för utvecklingen.
- Vissa modeller har ett tydligt fokus på vad som dokumenteras i varje steg, andra fokuserar på tiden.
- Vissa modeller fokuserar på händelser och sekvenser i användarnas arbetsuppgifter, andra fokuserar på informationsobjekt i verksamheten.

Beslutet kompliceras ytterligare av det faktum att man sällan samtidigt reviderar systemutvecklingsmodellen och projektstyrningsmodellen. Därför kan man lätt i

valet av en ny systemutvecklingsmodell lockas att välja en modell som är mer detaljerad vad avser projektstyrningsaspekter, som t ex DSDM.

Att ta ett helhetsgrepp på projektstyrning och systemutveckling hjälper inte, man har fortfarande ett alldeles för komplext beslut att fatta. T ex så hävdar DSDM konsortiet att RUP är en instans av DSDM samtidigt som RUP hävdar att denna kan anpassas till DSDM. Kort sagt så talar vi både äpplen och päron här.

I formuleringen och införandet av en användarcentrerad systemutvecklingsmodell blir det därför intressant att överväga huruvida en sådan modell skall omsluta både en systemutvecklingsmodell och en projektstyrningsmodell, om den skall vara något man pluggar in i en traditionell systemutvecklingsmodell för att få den användarcentrerad (i relation till projektstyrningsmodellen) eller om användarcentrering är ett synsätt som man skulle kunna lägga på all utveckling oavsett metod. Det är det sistnämnda som det ramverk som ISO 13407 presenterar, gör anspråk på att vara.

Resterande delen av detta kapitel behandlar projekt och projektstyrning ur ett mer allmänt perspektiv, ej specifikt anpassat till hur man bedriver användarcentrerade projekt. Avsnittet är författat av Inger Dahlgren på TietoEnator AB och är baserat på hennes mångåriga erfarenhet av att bedriva teknikutvecklingsprojekt. Det hade kanske hellre varit önskvärt att ha ett utkast till en projektstyrningshandbok med fokus på användbarhet och användarcentrerad systemutveckling, men något sådant finns inte utan det återstår att utveckla.

5.2 Vad är ett projekt?

Ordet projekt betyder förslag eller plan och kommer från latinet. Vi definierar det också som ett till omfattning och färdigtidpunkt bestämt arbete av engångskaraktär.

Ett projekt skiljer sig från löpande verksamhet och har t ex följande egenskaper:

- egen organisation
- eget mål
- beställare
- givna ramar med avseende på tid, kostnad och funktion
- fast start- och sluttidpunkt
- egen budget och redovisning

Det som kännetecknar ett lyckat projekt är framför allt nöjda beställare, nöjda användare och nöjda projektdeltagare.

5.3 Organisation

Ett projekt har en egen organisation som är skild från linjeorganisationens. För att organisationen skall göra nytta i projektet skall den spegla resultatet som skall levereras från projektet. Det betyder i ett systemutvecklingsprojekt att man skapar delprojekt/arbetsgrupper som utarbetar kravbeskrivning, lösningsbeskrivning, analysdokument, programspecar, tester, utbildning, användardokumentation etc. Det skall vara lätt att identifiera leveransen från respektive arbetsgrupp. Leveranserna från respektive arbetsgrupp/delprojekt bildar projektets totala leverans.

Följande grupperingar bör finnas i ett projekt:

I styrgruppen fattas beslut som inte projektledningen kan ta själv. Projektledningen fattar beslut inom givna ramar. Alla beslut utanför dessa ramar måste tas av styrgruppen. Styrgruppens ordförande bör vara projektets beställare. Styrgruppen är länken mellan beställaren/användaren och projektet.

Projektledningen leder arbetet i projektet. Den arbetar dels upp mot styrgruppen vilket innebär att den sitter föredragande i styrgruppen dels arbetar den inåt mot projektet vilket betyder att den har en arbetsledande funktion.

Det praktiska arbetet med att ta fram den produkt som skall levereras görs i delprojekt eller i mindre projekt av arbetsgrupper. Dessa grupper bör organiseras så att det blir tydligt vad som skall levereras från resp. grupp – design, tester, utbildning, användardokumentation etc.

Det resultat som tas fram i delprojekten/arbetsgrupperna bör förankras i beställaren/användarnas verksamhet. Det sker i referensgrupperna i arbetet tillsammans med användarna.. Det kan medföra att dokument som tas fram i projektet – kravbeskrivning, användardokumentation mm granskas av användarna. Det kan också bestå av arbete i fokusgrupper eller i andra former av användarmedverkan och användbarhetstester av prototyper/system.

Det finns också tekniska referensgrupper med deltagande från förvaltning och drift. Dessa grupperingar skall ta över resultatet från projektet och bör därför kopplas in i projektet så tidigt som möjligt.

Analysgruppen kallas in vid behov av projektledningen eller styrgruppen då man vill göra en genomlysning av hela eller delar av projektet. Analysgruppen består av deltagare som ej arbetar i projektet. Den måste kunna titta på projektet med fräscha ögon som inte är påverkade av projektarbetet. Se nedan under projektgranskningar.

Organisationen av ett projekt är inte statisk. Den skall förändras under tiden beroende på hur innehållet i projektet förändras. Det är viktigt att förändra

organisationen samt att bemanna den på ett sätt som gagnar arbetet i projektet, se mer nedan.

5.4 Mål

Ett projekt har ett eget mål som är skapat för just detta projekt. Det är viktigt att man skiljer på projektmål och effektmål.

Projektmålet är det resultat som projektet skall uppnå. Det kan vara att ta fram en produkt, bygga ett hus eller utveckla ett IT-system.

Kriterierna för ett bra mål är:

- tydligt
- mätbart
- realistiskt
- tidsbegränsat

Det får inte råda något tvivel om vad vi skall uppnå i projektet. Målet skall därvidlag vara konkret och påtagligt. Dessutom skall man kunna mäta i efterhand om målet har nåtts. Även om målet bör ha ett visst mått av utmaning i sig så bör det också vara realistiskt. Ett mål skall dessutom alltid vara tidsbestämt. Det skall framgå när det skall vara uppnått.

Ett exempel på ett mål kan vara: Den 1 januari 2000 är affärssystemet XX i drift i vår organisation.

Effektmål är inte detsamma som projektmål. Effektmålet beskriver nyttan som organisationen förväntar sig av det som projektet tar fram dvs effekterna av projektmålet. Om ett projekt har som mål att utarbeta ett system så är effektmålet den nytta som systemet är tänkt att åstadkomma i organisationen.

Exempel på effektmål är: Projektet bidrar till vår organisation genom att kostnaderna förväntas minska med YY kr under en femårsperiod/intäkterna förväntas öka med ZZ kr.

Man måste tidigt i projektet bestämma vad som är prioriterat – styrningen på tid, kostnad eller funktionalitet. Vad är viktigast i projektet – att vi blir klara enligt tidplan, enligt budget eller enligt de av beställaren/användarna framställda kraven. Det är självfallet klart att man skulle vilja prioritera alla tre parametrarna. Ofta kommer man dock till en punkt då man inte kan tillgodose dem alla utan måste prioritera. I systemutvecklingsprojekt förr var oftast funktionaliteten viktigast, numera har tidsstyrningen i projekt blivit vanligast. Det är viktigt att snabbt få ut den nya produkten på marknaden. I realiteten

handlar det ofta om en kombination – lite mera tid, men lite mindre funktionalitet. Man måste kanske också prioritera mellan funktionerna.

5.5 Affären vs projektet

Ett projekt har alltid en beställare. Den personen kommer oftast från affärsverksamheten antingen inom den egna organisationen eller från en extern organisation. Beställaren finansierar oftast projektet. Projektet görs på uppdrag av beställaren. Det är viktigt att man i organisationen håller isär projektets verksamhet från affärsverksamheten. Projektet måste få leva sitt liv med sin beslutade budget oavsett vad man beslutar inom affärsverksamheten. Om affärsverksamheten får påverka projektet löpande kommer arbetet i projektet att bli väldigt ryckigt.

Projektet berörs dock av vad som händer på affärsnivån. Nya krav på systemet kommer oftast från affärs/användarsidan. Omvärlden förändras vilket påverkar projektet. Det resultat som tas fram i projektet kanske inte längre är relevant. Då måste affärssidan besluta om att kanske lägga ner projektet. Dessa beslut når projektet via styrgruppen som är länken mellan affärssidan och projektet.

5.6 Planering

5.6.1 Tid

Ett projekt kan i princip vara hur långt eller hur kort som helst. Det viktiga är dock att man från början bestämt när det skall ta slut. Ett sätt att se till att man alltid avslutar ett projekt är att ha en leverans. Utan leverans är det lätt att ett projekt aldrig tar slut. Det är också viktigt att man bestämmer vilken leverans som skall avsluta projektet. Det är lätt att lägga till den ena leveransen efter den andra vilket resulterar i att projektet aldrig tar slut. Det är också viktigt att särskilja löpande verksamhet från projektverksamhet. Löpande verksamhet har aldrig ett slut varför ett sådant »projekt« aldrig kommer att kunna avslutas.

Av praktiska skäl kan det dock vara bra att hålla ett projekt så kort och litet som möjligt. Det blir då lättare att överblicka och hantera. Avgörande för hur stort och långt ett projekt kan vara är dock inte själva tidsaspekten utan innehållet. Ett projekt kan inte vara mindre än att man kan identifiera det mål som man sedan skall leverera.

Det är dock klokt att dela in projektet i etapper där varje etapp avslutas med en leverans. Varje leverans bör därvid vara minsta möjliga del av det totala resultatet som kan användas av kunden. Ju fler etapper och därmed delresultat desto bättre.

Faser

Man kan i ett projekt identifiera tre faser

1. Förberedelser
2. Genomförande
3. Avveckling

Under förberedelsefasen planerar man själva genomförandet i projektet. Det innebär att man genomför följande aktiviteter:

- idé och målformulering
- tidplanering
- resursplanering
- kalkylering
- bestämmer organisationen
- bemannar projektet

Genomförandefasen kännetecknas av att det är den del av projektet då arbetet med att utarbeta projektets resultat dvs att ta fram produkten, systemet etc.

Under avvecklingsfasen ser man till att projektet avvecklas på ett snyggt och prydligt sätt att personalen avvecklas och att projektet skriver erfarenheter.

För att sköta arbetet inom de tre faserna på effektivt sätt krävs en bra tidplanering.

Tidplanen innehåller de aktiviteter som skall genomföras i respektive fas samt start- och sluttidpunkter för respektive aktivitet. Tidplanen anges i kalendertid.

Tidplanen måste göras på tre nivåer:

Projekttidplan – för hela projektet

Detaljtidplan – för delar av projektet

Personlig tidplan – för respektive medarbetare i projektet

Det är viktigt att alla som arbetar i projektet har en egen personlig tidplan där det framgår vilka aktiviteter man är ansvarig för.

Tidplanen måste kompletteras med en resursbehovsplan. I den anger man för resp. aktivitet hur många timmar eller hur många personer som behövs. Resursbehovsplanen är underlag för bemanningsplanen.

Den iterativa utvecklingsprocessen kännetecknas av att man kontinuerligt utvärderar ändamålsenligheten och användbarheten i sina lösningsförslag. Det här innebär också att man måste kunna gå tillbaka och kasta eller förändra

lösningsförslag och även de analyser som ligger till grund för lösningsförslagen. Det nya eller modifierade lösningsförslaget utvärderar man återigen. Så arbetar man tills man uppnått ett i förväg bestämt mål som kan gälla t ex användbarheten i lösningen.

Det för med sig att tidplaneringen blir densamma som i ett traditionellt utvecklingsprojekt bara med mycket kortare aktiviteter och därmed ett snabbare förlopp. Om man i traditionell systemutveckling gör all kravbeskrivning först för att därefter fortsätta med all analys, design och konstruktion etc så genomgår man i den iterativa processen samma utvecklingssteg men man gör alla stegen flera gånger i en iterativ snurra vilket betyder att kravbeskrivningen t ex förändras efterhand som arbetet fortskrider. Aktiviteterna kommer också att ligga parallellt i större utsträckning än vid traditionell systemutveckling då aktiviteterna oftast är sekventiella.

5.6.2 Kostnad

Ett projekt har egen budget och redovisning. Budgeten består endast av kostnader. Investeringskalkyler bör göras innan man startar ett projekt för att bedöma lönsamheten för produkten som tas fram inom projektet. Den utarbetas dock på den affärsmässiga sidan inom organisationen. Intäktssidan i kalkylen kommer aldrig att beröra projektet eftersom den inträffar efter det att projektet är klart. Det är endast kostnadsdelen av kalkylen som är projektets budget.

Hur budgeterar man?

Ett projekts budget bör bestå av följande delar:

- aktivitetsbaserat arbete t ex kostnader för programmering, test
- ledning och administration t ex kostnader för projektledning
- inköp t ex kostnader för inköp av datorer, programvara
- post för oförutsedda utgifter t ex befarade kostnader för mertid beroende på att programmeringen kanske tar längre tid än vad som är budgeterat under aktivitetsbaserat arbete

5.6.3 Funktion

Beställaren av ett projekt bör alltid utarbeta någon form av kravbeskrivning där beställarorganisationens krav på produkten anges. Kravbeskrivningen utgör underlag för arbetet i projektet.

För att tillse att kraven hålls aktuella måste projektet fortlöpande ha en kravdialog med beställaren/användarna. Omvärlden förändras ständigt och påverkar resultatet som skall tas fram i projektet. Dessutom får man större

kunskap om resultatet vartefter projektet fortskrider vilket också påverkar kraven. Dessa måste därför nästan alltid ändras under resans gång. Kravdialogen med beställaren/användarna kan bestå av demonstrationer, prototyper, regelbundna möten där nya krav diskuteras, fokusgrupper eller andra grupparbetstekniker etc.

Det är viktigt att se till nyttoaspekten inom projektet. Beställaren skall alltid ha nytta av resultatet från projektet. Om inte projektet kan bidra till organisationens verksamhet eller inte stödjer användarna i deras behov bör man ifrågasätta det mål som projektet har att utföra och förändra det eller lägga ner projektet.

Om man kan identifiera fler nyttor med resultatet som tas fram i projektet kan det vara nödvändigt att prioritera de som är viktigast och lägga dem först i utvecklingsarbetet.

Ett projekt kan också behöva prioritera mellan tid, kostnad och funktion/nytta. Om en produkt måste ut på marknaden snabbt måste tiden som projektet tar prioriteras framför kostnad och funktion. Om nyttan är den viktigaste parametern måste projektet få ta den tid och kostnad det behöver så att alla definierade funktioner finns med i produkten.

5.7 Bemanning av projekt

Ett projekt kan bemannas antingen från den egna organisationen eller med externa resurser. Det som framför allt utmärker ett projekt är att man sätter samman resurserna från olika delar av organisationen.

Användare måste finnas med på alla nivåer inom projektet. I styrgruppen bör beställaren sitta ordförande. Minst en representant för slutanvändarna bör också vara med i styrgruppen. I arbetsgrupperna måste användarna vara med i allt arbete utom det som är direkt IT-relaterat såsom design och konstruktion. Användarna måste alltså vara med i beskrivandet av verksamhetsprocesserna, kraven/nyttan, utvärderingarna, testerna, användardokumentation och utbildning etc kontinuerligt under projektets gång.

Det är viktigt att projektledaren inser vikten av användarmedverkan i projektet. Det är projektledaren som måste se till att kravdialog hålls löpande med beställaren/användarna samt att användarnyttan inte glöms bort utan hålls levande under hela projektets gång.

Man väljer resurser baserat på två parametrar:

- kompetens
- lämplighet

Det är viktigt att personalen i projektet har rätt kompetens eftersom projekt ofta måste arbeta snabbt och effektivt. Det kan dock vara svårt att alltid få de bästa resurserna. Då får man inte glömma att det alltid finns möjlighet att utbilda mindre erfaren personal.

Den kanske allra viktigaste parametern när man skall bemanna projekt är dock att se till att man får personer med personliga egenskaper som gör dem speciellt lämpade att arbeta i projektet. Det kan röra egenskaper som samarbetsförmåga, flexibilitet och motivation. Dessa egenskaper har ofta avgörande inverkan på projektets förmåga att prestera ett bra resultat.

Det är viktigt att projektet får rätt resurser. Därför skall man lägga ner möda på att engagera rätt personer. Man bör ha ett bemanningsförfarande som påminner om ett vanligt anställningsförfarande.

Man bör alltså aktivt söka rätt personer, »anställa« dem till projektet och se till att de får en ordentlig introduktion till projektet. Under »anställningsfasen« bör man berätta om projektet så att personen i fråga har möjlighet att sätta sig in i projektet och ta ställning till om detta är i linje med den egna utvecklingsinriktningen.

När man tar ställning till om man vill arbeta i ett projekt är det viktigt att man ser på arbetsuppgifterna som ett personligt åtagande. Det innebär att jag inte kan åta mig andra uppgifter parallellt om det inte är helt klart att jag vet att jag klarar av båda samtidigt. Ett åtagande i ett projekt måste jag genomföra även om det dyker upp andra uppgifter som jag hellre vill göra. Detta synsätt måste prägla samtliga deltagare i projektet. Det personliga åtagandet bör bekräftas i en skriftlig uppdragsbeskrivning. Den innehåller uppgifter om de aktiviteter som skall utföras, när i tiden de skall göras, hur mycket mantid som är tänkt åtgå samt när aktiviteterna skall vara klara för leverans. Till uppdragsbeskrivningen bifogar man den personliga tidplanen.

När projektet är avslutat är det också viktigt att alla personal återlämnas till linjeorganisationen och att det då finns nya arbetsuppgifter för dem där.

5.8 Lokalisering av projekt

Projekt bedrivs företrädesvis i beställarens/användarnas organisation. Där blir närheten till användarna störst och till den organisation som skall använda resultatet av projektet. På det viset får projektet mer input än om det bedrivs hos leverantören. Arbetet tillsammans med användarna blir mer kontinuerligt och användarna känner sig mer delaktiga.

För att uppnå en bra informationsspridning är det också önskvärt att alla i projektet sitter samlat på ett och samma ställe.

5.9 Uppföljning

När man väl gjort en bra tidplan och då ända ner på personlig nivå är det viktigt att den följs upp på den personliga nivån också.

Projektet bör ha regelbundna uppföljningsmöten då man bl a stämmer av tidplanerna.

Om möjligt skall hela projektet deltaga i projektmötena. Vid uppföljningen bör man fokusera på återstående tid och arbete. Den tid som är nedlagd och de arbetsuppgifter som är genomförda skall noteras men fokus skall ligga på återstående tid och aktiviteter.

Det är i detta sammanhang nödvändigt att samtliga i projektet har samma definition av begreppet »klar«. I ett systemutvecklingsprojekt kan »klar« betyda bl a:

- färdigprogrammerad men ej testad
- programtestad men ej systemtestad
- systemtestad men ej acceptanstestad
- acceptanstestad men ej levererad
- levererad

Förseningar uppstår i de flesta projekt. Det är då viktigt att man analyserar förseningen så att man vet varför den har uppstått – och därvid förhindrar att den uppstår igen - samt att man utarbetar ett åtgärdspaket för omedelbart genomförande. Det är nödvändigt att man försöker »komma upp på banan igen«.

Åtgärdspaketet kan t ex bestå av

- övertid
- förkortad semester
- helgarbete

men även

- minskad funktionalitet
- fler resurser

Att sätta in fler resurser är inte alltid så lätt som det kan verka. Nya medarbetare tar tid att lära upp och därför bör man vara försiktig med att sätta in nya resurser i kritiska skeden i projektet. Det kan kosta mer än det smakar. I det läget är det oftast bäst om befintliga medarbetare i projektet kan arbeta mer.

5.10 Riskanalys

Att löpande göra riskanalyser i ett projekt är att kvalitetssäkra resultatet. Man bör med jämna mellanrum hålla seminarier i projektet där syftet är att identifiera hot och risker mot projektet. Det är viktigt att man tidigt kan upptäcka risker som äventyrar vår förmåga att leverera rätt resultat från projektet. Riskanalyser görs inom projektet med deltagare från projektet.

En riskanalys görs lämpligast som ett halvdagsseminarium där man genomgår följande steg:

- identifiering av risker enskilt
- sammanställning av samtliga risker i sk brainstorming-anda dvs inga risker är för små. Alla risker måste tas fram och presenteras.
- bedömning av riskerna m a p sannolikhet att de inträffar samt effekten på projektet eller organisationen om de inträffar. Här sållas de små riskerna automatiskt bort.
- identifiera åtgärder för att eliminera riskerna
- budgetera och genomföra åtgärderna

Risker som ofta återkommer i projekt är:

- nyckelpersonsberoende
- användarna har ej tillräckligt med tid för projektet
- sjukdom
- omvärldsförändringar
- minskad budget

5.11 Projektgranskning

Man bör dessutom göra projektgranskningar i alla projekt. Syftet med dem är också att kvalitetssäkra resultatet men då med fokus på hur vi arbetar i projektet, är vi rätt bemannade, har vi bra tidplaner som projektet kan arbeta efter, finns det en bra utvecklingsmodell, finns det ett uttalat mål för projektet, vet alla vad som skall levereras och när etc. Projektgranskningar bör göras av en extern granskare.

Problem som ofta identifieras vid en projektgranskning är:

- kravbeskrivningen är ej tillräckligt genomarbetad
- lösningsbeskrivningen är ej förankrad hos användarna
- projektet följer ej utvecklings- och projektstyrningsmodell
- projektdeltagarna har inte personliga planer

- uppföljning av projektet görs ej på personlig nivå

5.12 Ledarskapet i projekt

Om man behöver en utvecklingsmodell för att ta fram rätt produkt i projektet och en styrningsmodell för att arbeta på rätt sätt så behövs det ledarskap för att få människorna i projektet att arbeta för att nå det gemensamma målet.

Grunden för ett lyckat projekt är:

POSITIV MÄNNISKOSYN. Alla i projektet, IT-folk och användare, måste ha en förtroendefull inställning till varandra. Den bygger på ömsesidig respekt för varandras kompetens och en insikt om att alla behövs och är viktiga för att projektet skall lyckas. Ingen kan allt i ett projekt men alla kan något och det behövs samarbete där var och ens förmåga måste användas för att uppnå ett bra resultat.

SAMFÖRSTÅND. I ett projekt måste båda parter ha respekt för varandras behov. Både beställare och leverantör, uppdragsgivare och uppdragstagare, projektledare och projektmedlem, utvecklare och användare skall känna att de har något att vinna i projektet. I ett projekt där det finns förlorare åstadkommer man inte bra resultat. Det måste råda ett »vinna-vinna-förhållande« i alla situationer.

ÅTAGANDE. För att arbetet skall dels bli stimulerande för de som arbetar i projektet dels bli hanterbart för projektledaren måste man huvudsakligen ha ett ledarskap som är delegerat. Projektdeltagarna måste ta ansvar för sina resp. aktiviteter och genomföra dem självständigt under eget ansvar.

Grunden för alla medarbetares deltagande måste vara det personliga åtagandet. Alla som arbetar i ett projekt måste åta sig att genomföra de aktiviteter man kommit överens med överställd chef – i ett projekt blir det oftast projektledaren.

6

Design och systemutveckling i praktiken

Baserat på flera år av »sammandrabbningar« med såväl användare som utvecklare i olika IT-utvecklingsprojekt har vi alltmer börjat intressera oss för hur systemutvecklingsprocessen *egentligen* går till samt hur systemutvecklare bedriver sitt arbete när man gör användargränssnittsdesign.

6.1 Hur arbetar gränssnittsutvecklare egentligen med design

Oftast är det systemutvecklare som arbetar med och har ansvar för utvecklingen av gränssnittet. Detta ansvar är oftast i och för sig outtalat, vilket kanske följande anekdot kan visa på:

» *Vid ett sammanträffande med en organisations ansvariga för utvecklingen av användargränssnittet i en större utvecklande organisation frågade vi vem som var ansvarig för layout och form, vem som fattade designbeslut och hur detta gick till. Alla*

> *de ca. 25 närvarande personerna tittade generat ned i bordet till dess att en tittade upp och sade att 'Det är Björn' och pekade tvärs över bordet. Björn tittade förvånat upp och sade 'Jag? Vad menar du egentligen?' varpå den andre svarade 'Tja, du har ju alltid sagt att du tyckte om att rita i skolan.'«*

Detta är ganska symptomatiskt för hur organisationer har hanterat frågan med gränssnittsdesign. Gränssnittet bara uppstår utan att det är tydligt för någon att ett antal designbeslut har fattats. En utvecklare kan pekades ut som mer ansvarig eftersom han hade ett mer konstnärligt sinne och »var ganska duktig i teckning i skolan«. Han kunde konsulteras vad gäller de få estetiska frågor som kommer upp på bordet (bakgrundfärg, etc). Väldigt få estetiska frågeställningar kommer dock någonsin upp.

Detta är på intet sätt en svensk företeelse vilket man bland annat kan läsa mycket målande i boken »*The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity*« av Alan Cooper [Cooper, 1999].

Vi ville undersöka hur dessa förhållanden faktiskt såg ut i Sverige och utförde därför ett antal observationsintervjuer av systemutvecklare, gränssnittsutvecklare och hur deras arbete med utformningen av gränssnittet går till. Huvudsakligen intervjuade vi personer på in-house utvecklingsbolag, dvs. bolag som utvecklar system som skall användas inom den egna koncernen och huvudsakligt fokus var statliga myndigheter (det finns dock skäl till att tro att situationen inte skulle vara mycket bättre på andra utvecklande bolag). Typiskt var systemutvecklarna av två huvudsakliga karaktärer:

- antingen en före detta verksamhetsmänniska inom organisationen som genom intresse, internutbildning och medverkan i olika projekt lärt sig hantverket,
- eller också en person som har sitt första arbete efter en systemvetarutbildning, eller en mindre datautbildning. Det är mycket ovanligt att se mer erfarna utvecklare som har arbetat på ett flertal olika arbetsplatser.

Orsaken till detta är förmodligen flera; en statlig myndighet associeras inte automatiskt till att vara en ny och innovativ arbetsplats där man får lära sig mycket och utvecklas samt får bra betalt, trots att flera av dessa organisationer insett sig behöva erbjuda mer konkurrenskraftiga löner i det senaste. Sålunda bedrivs verksamheten mer förlitande på s. k. mentorer (externa konsulter) som står för de huvudsakliga lösningarna, och de fast anställda

systemutvecklarna, som på ett betydligt mindre effektivt sätt arbetar med problemen.

SYSTEMUTVECKLING ÄR EN PROBLEMLÖSNINGSPROCESS. Ingen systemutvecklare arbetar uttalat för att göra användbara system, man arbetar i termer av problemlösning som har uppnått som en normal konsekvens av en strävan att dela in arbetet i små hanterbara delar med en tydlig och uppnåelig målsättning. En utvecklare tilldelas eller tar för sig ett visst problem. Att lösa problemet innebar att skapa en snutt programkod som löser uppgiften. Kvaliteten i arbetet låg i att göra denna programkod så effektiv som möjligt. Det faktum att denna program kod skapar ett gränssnitt som har ett visst utseende och att detta var pga. Deras kodning kom som en överraskning för många av utvecklarna. Ansvaret för utseendet hänger i någon bemärkelse i luften. Ett av problemen är att det inte finns något i metoderna som stödjer utvecklaren i formuleringen av de problem som man vill lösa.

GRÄNSSNITTEN UPPSTÅR. Att gränssnittsdesignen är en kreativ process kan tyckas vara självklart, men för de som är djupt inblandade i systemutvecklingsarbete ser man inte att det är kreativt arbete som pågår.

MÖTEN. Bortsett från TV är Sveriges största konsument av ineffektiv tid *möten*. Möten arrangeras alldeles för ofta med alldeles för många deltagare utan att det egentligen är nödvändigt. Slående med möten är oftast att alla som över huvud taget skulle kunna komma att ha något som helst intresse av någon del av mötet blir kallade och tvingas sitta med och kommer därför också att ha åsikter om alla möjliga frågor, som egentligen inte ligger inom deras område. Folk känner också att man bör vara med på möten eftersom det kan vara svårt att få information om viktiga händelser på andra sätt. Därför åtgår en mycket stor del av de anställdas tid till att sitta på möten, möten som ofta kan ha 20 deltagare eller mer. I det perspektivet kanske följande riktlinjer kunde vara användbara:

- **ANTAL DELTAGARE**. Undvik möten med mer än 6–8 deltagare.
- **ANGELÄGENHET**. Behandla de punkter som angår alla först, entlediga de som inte behövs fortlöpande. Det är inte oartigt att släppa iväg de som inte direkt kan ha bidragande åsikter, snarare är det oartigt att uppta folks tid utan att de bidrar. Det finns en stor mängd professionella mötesdeltagare som alltid har åsikter och ventilerar dessa – även i de fall frågorna inte angår dem.
- **FÖRBEREDELSER**. Se till att du är förberedd inför ett möte och ställ även detta krav på andra medverkande. Att skriva protokollet i förväg är ett sätt att tidigt gå igenom och lista de beslut som måste fattas.

- **DISKUSSIONER**. Fokusera på problem och på frågor som kräver diskussion. Avrapportering av saker som går bra är förvisso trevligt men kan uppnås på annat sätt.
- **STOPPA MÖTESGENERATORER**. Det är ofta lätt hänt att problem som kommer upp kan tendera att generera nya möten. Undvik detta. Tag hellre tag i problemen med en gång och föreslå potentiella lösningar.
- **FATTA BESLUT**. Beslutsoförmåga är mycket vanligt förekommande, troligen av anledningen att man vill sträva efter konsensus bland de inblandade parterna. Fatta alltid beslut och undvik bordläggning. Det är precis lika enkelt att riva upp ett felaktigt beslut som att fatta beslut efter flera olika diskussioner.
- **UNDVIK CHEFSMEDVERKAN**. Det finns en spridd uppfattning att man behöver låta cheferna delta för att berättiga vissa beslut. Fråga alltid om detta verkligen är befogat och om det verkligen bidrar till projektet.

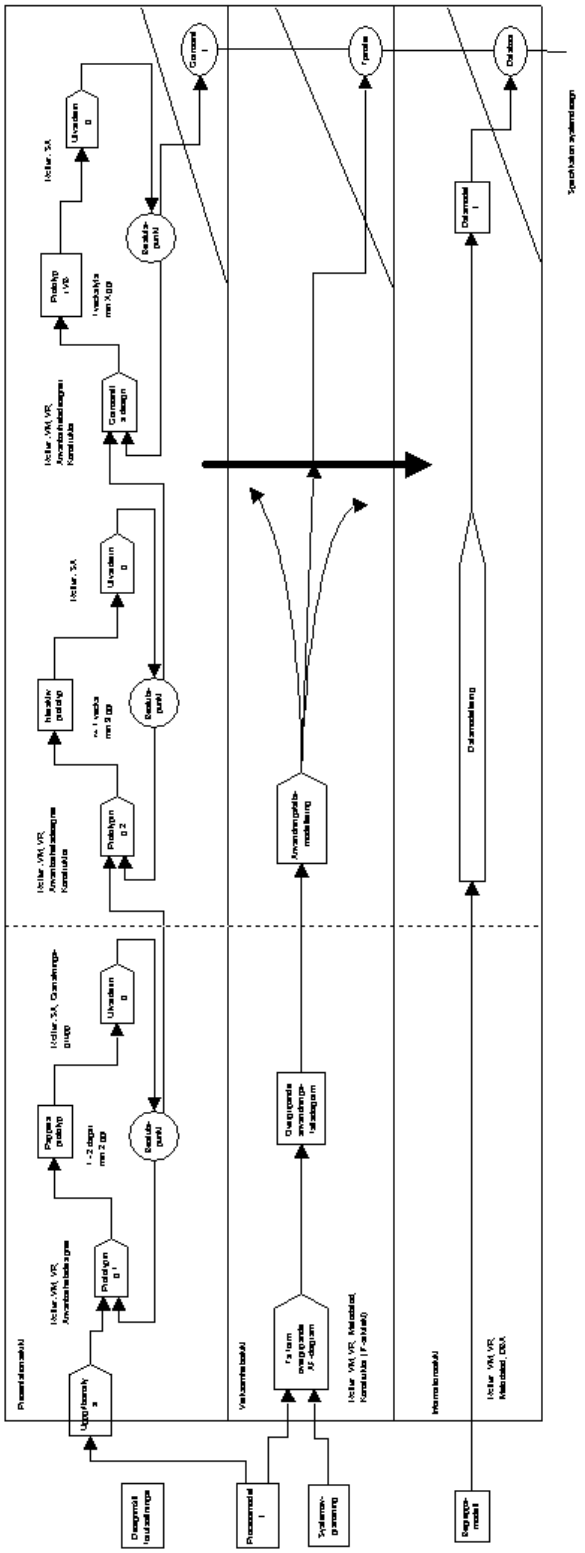
6.2 Beskrivning av en systemutvecklingsmodell – ett praktikfall

Detta avsnitt tar upp RSVs systemutvecklingsprocess som ett praktikfall. Dagens systemutvecklingsmodell hos RSV kan ses som en variant på ett vattenfall:

- Förstudie
- Analys
- Utformning
- Konstruktion
- Driftsättning

Den är formad efter de erfarenheter som organisationen samlat på sig efter decennier av systemutvecklingsprojekt. Om man ser den i skenet av hur användarcentrerad systemutveckling bör gå till, kan vi konstatera att den saknar en hel del. Detta var också utgångspunkten vid det arbete som bedrivits för att titta på hur en framtida användarcentrerad systemutvecklingsmodell skulle kunna se ut hos RSV.

6.2.1 Re-modellerad utvecklingsmodell

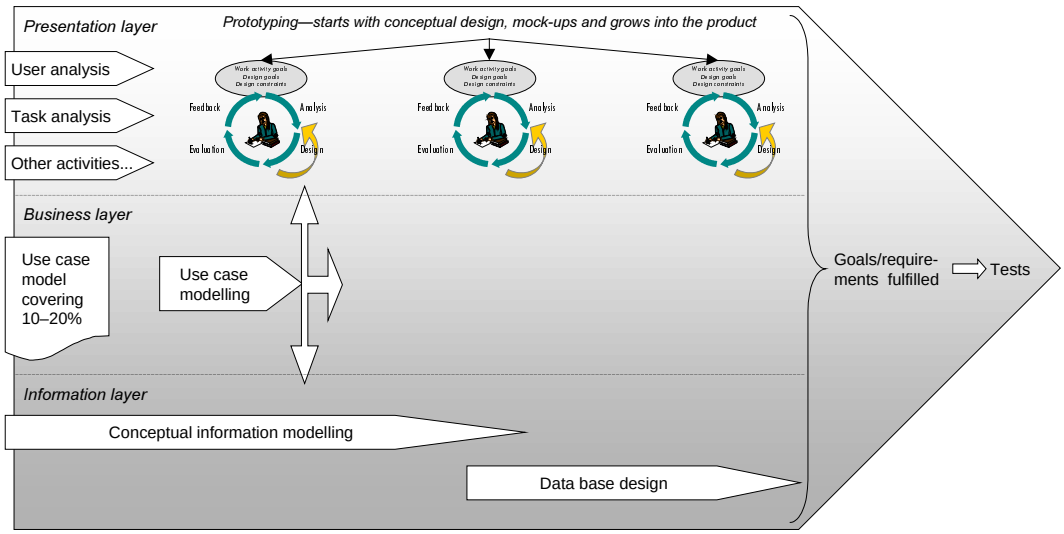


Figur 21. Tänkt användarcentrerad systemutvecklngsmodell hos RSV.

Ovanstående figur är det schematiska resultatet av det modelleringsarbete vi genomfört tillsammans med RSV. Den visar på en tänkt användarcentrerad systemutvecklingsmodell hos RSV. Den är naturligtvis inte heltäckande men visar ändå på viktiga delar i en användarcentrerad modell. Man bör lägga märke till att modellen börjar där själva systemutvecklingsarbetet inleds och alltså inte innehåller någonting om verksamhetsutvecklingen som sådan. Modellen slutar sedan innan installation och införande.

Den är uppdelad enligt de tre skikt som RSV använder för sin systemarkitektur. Modellen är inte på något sätt färdig utan avses främst att tjäna som ett underlag för ytterligare diskussioner.

I figuren nedan (*Figur 22*) har vi gått vidare och börjat »renrita« modellen.



Figur 22. Variant på användarcentrerad systemutvecklingsmodell.

Vi kommer att jobba vidare med dessa modeller för att förhoppningsvis senare kunna presentera en mer komplett användarcentrerad systemutvecklingsmodell för RSV.

6.3 Hur skulle RSVs systemutvecklingsmodell kunna utvecklas

Oavsett vilken systemutvecklingsmodell en organisation väljer att följa är det viktigt att man gör den till sin egen.

Vad avser användbarhet och användarcentrerad systemutveckling bör organisationens användarcentrerade modell vara organisationens egen och formuleras i samverkan mellan de som skall komma att använda den och specialistkompetensen inom ACSU. Så har det skett på RSV i projektet »Användarcentrerad systemutveckling på RSV« i vilket personal från avdelningen för Människa-datorinteraktion vid Uppsala universitet har deltagit

som experter på ACSU i den modellering som RSV har gjort av sin egen systemutvecklingsverksamhet.

Det finns många definitioner på ACSU i vetenskapssamhället och vi blev tvungna att formulera en egen modell för RSV som skulle inbegripa de aspekter som vi alla tyckte var viktiga att fånga för just RSV. Den modell som vi enades om under diskussionerna på RSV är inspirerad av ISO 13407 (Human Centred design process of interactive systems) samt de riktlinjer för att göra användbara system som publicerats av Gould & Lewis på IBM (1985).

Denna definition enades RSVs ACSU-grupp om:

Användarcentrerad systemutveckling – en systemutvecklingsprocess som eftersträvar användbara system genom:

- **Arbetet skall tidigt styra utvecklingen**, för att undvika utarmning och onödig styrning. Tidig fokus på användarna och deras uppgifter. Designern måste förstå vilka användarna kommer att vara, deras kognitiva, beteende- och attitydmässiga inställning, samt arbetets karaktäristik. Vettig allokering av funktioner mellan användare och system.
- **Aktiv användarmedverkan redan i projektets början**, i analys-, design-, utvecklings- och utvärderingsarbetet. Urvalsprocessen och kompetensen är viktig. Interaktiv design: typiska användare måste bli en del av utvecklingsteamet från första början. Användarmedverkan av både:
  - Verksamhetsexperter (kontinuerligt igenom ett utvecklingsprojekt).
  - Slutanvändare (för ut testning av olika designresultat).
- **Tidig användning av prototyper** för att testa och utveckla designlösningar.
- **Kontinuerlig iterering av designlösningarna.** En cyklisk process av design, utvärdering och omdesign skall upprepas så ofta som det är möjligt. Empirisk mätning. Experiment med prototyper med vilka riktiga användare gör riktiga arbetsuppgifter i syfte att deras reaktioner och attityder skall observeras, samlas in och analyseras.
- **Tvärvetenskapliga designteam.** Inkludera en användbarhetsdesigner (avsnitt *8.2 Användbarhetsdesignern*) i processen.
- **Integrerad design** (inom RSV ofta uttryckt som full AU-bredd)**.** Kontinuerlig utveckling av system, verksamhets, hjälp, utbildning, organisation, etc. i utvecklingsarbetet.

Ett sådant systemutvecklingsprojekt är en del av en utvecklingskedja där drift och utveckling av nya versioner även är en del av helheten.

Följande fem åtgärder ansåg vi vara särskilt viktiga att beakta för att RSV skulle få en bra fungerande ACSU-modell:

1. **Iterativ design**
   En av de centrala punkterna i ACSU är att designprocessen måste vara *iterativ*. Det är dock svårt att förstå vad man de facto menar med en *iterativ* process. Att en gränssnittsutvecklare stöter på ett problem och ringer upp en användare och frågar sig till råds för att därefter fortsätta sin programmering är på inget sätt iterativt. Iterativ utveckling måste inbegripa att man utför en mängd iterationer där varje iteration innebär:

   a) en ordentlig analys av användarens uppgifter och sammanhanget

   b) en prototyp-utformningsfas, och

   c) en dokumenterad användbarhetsutvärdering av designprototypen som faktiskt producerar utvärderingsresultat som måste adresseras i den kommande processen.

2. **Riktlinjer för användarmedverkan och urval**
   Organisationen visade ett utryckligt behov av riktlinjer för användarmedverkan, dvs. var, när och hur skall användarna delta i utvecklingsarbetet? Dess bör inbegripa frågeställningar som:

   - Skall de (kan de?) delta i sin egen arbetsmiljö, kan man låta systemutvecklarna resa till användarna snarare än tvärtom? Kan användarna delta per distans ifrån deras egen arbetsmiljö, kan man t ex låta dem använda prototyper på distans (t ex via Internet) och samtidigt besvara enkäter som frågar dem om användbarheten?
   - Kan de delta med deras eget språk, utförs modelleringarna med användarnas terminologi eller tenderar de att få lära sig systemutvecklarnas »fikonspråk«.
   - Skall de delta i analysfasen? I designfasen? I utvärderingsarbetet? I kodningen? Det är mycket vanligt att stora förändringar sker som påverkar utformningen efter det att utformningsfasen är avgjord och man kommit in i sjöälva kodningsfasen.

   Det behövs riktlinjer som behandlar hur man hanterar återkopplingen från användarna:
   - Samla in och dokumentera alla användarsynpunkter.
   - Tag ställning till alla synpunkter och besluta att:

  - Ändra i enlighet med kommentarerna.
  - Inte ändra och för att behålla användarnas förtroende är det då mycket viktigt att man meddelar användarna vilket beslut man tagit och varför i enlighet med de kommentarer som användarna lämnat.

Det behövs riktlinjer för hur man väljer ut användarna:

- **Slumpmässigt urval.** Att slumpmässigt välja ut användare ger förmodligen ett bra tvärsnitt av användarpopulationen, men knappast en bild som visar på den stora variation som användarna kan representera.
- **Maximera skillnader.** Detta kan alternativt vara en bra urvalsstrategi för att finna så olika användare som man kan tänka sig att finna, t ex att ta en generalistanvändare från skattemyndigheten i Blekinge (som måste behärska många olika typer av ärenden) och ta en specialistanvändare från Skattemyndigheten i Vällingby (som är specialiserad så till den grad att man bara hanterar ärenden av en viss typ för en viss specifik bransch.
- **Flexibla och öppna användare.** Vilka attityder som den valda användaren har till utvecklingsarbete, hur förändringsbenägen man är och i vilken utsträckning man kan lyfta sig att se sin egen arbetssituation från ovan är viktiga aspekter för att klara av att bli en bra användarrepresentant.
- **Social kompetens.** En mycket stor del av den tid som användarna lägger ned på deltagande i projekt går åt till möten och modelleringar. Det är då viktigt att man får en grupp som fungerar bra tillsammans och som man kan arbeta bra tillsammans med.
- **Representativitet.** Känner användarna att de representerar en grupp användare, eller är de där för att föra fram sina egna åsikter. Vilket uppdrag man har kan i stor utsträckning styra resultatet av ens medverkan.
- **Frivillig?** Är användarna frivilligt med i utvecklingsprojektet eller har de kommenderats dit?
- **Anonymitet?** Ger man användarna möjlighet att framföra kritik utan att det slår tillbaka på dem själva, ges de verkligen en möjlighet att kritisera system eller arbetssätt.
- **Gruppstorlek.** Det är välkänt att stora gruppstorlekar inte befrämjar effektivt arbete. 5–7 personer är en optimal storlek på en

grupp. Man kan undvika att få med alla intressenter i möten genom att tydligare specificera syfte och status med träffen och att tydligt stänga ut de som vill dit för att bevaka en position snarare än att bidra till utvecklingen.

- **Gruppkarakteristik.** Hur gruppen är utformad i termer av könsfördelning, maktrelationer, andel systemutvecklare kontra användare, etc. kan även detta ha avgörande betydelse för arbetet.
- **Användarna i majoritet.** Att ha med flera användare som kan stödja varandra i arbetet är mycket värdefullt för att skapa en kreativ stämning. Att totalt sett i projektet låta användarrepresentanter i antal dominera över »tekniker« är också rekommenderbart.
- **Positiv samverkanston**. Att undvika att framföra kritik utan att utnyttja regelverket för brainstorming kan vara användbart, kan man framföra positiva konstruktiva förändringar, etc. så kan man ofta stimulera till ett kreativt klimat.

Slutligen är det viktigt att poängtera att man bara för det att man en gång har jobbat i verksamheten så är man ingen typisk användare. En användare som hyrs in på mer en halvtid i ett projekt har ett bäst före datum på några veckor. Därefter tenderar man snabbt att falla in i en roll i vilken man blir en förespråkare för projekten. »Inlåningar« som arbetar heltid i projekt brukar vi därför att kalla för domänexperter för att skilja dessa från de användare som vi tillfälligt tar in för t ex. utvärderingar.

3. Prototyping

Att konstruera gränssnittet genom att använda prototyper har i en annan sektion i rapporten beskrivits utförligt. Det är viktigt för en fungerande iterativ process att man faktiskt kan arbeta för att prototyptekniker blir vanligt förekommande i arbetet. Därför nämner vi här några idéer till hur dessa arbetsmetoder skulle kunna utvecklas i framtiden:

- **Tidigare prototyping.** Allt som oftast börjar man med aspekter som relaterar till utformningen av gränssnittet alldeles för sent i utvecklingsarbetet. Många av de saker som man vill åstadkomma är inte möjliga för att man har målat in sig i hörn i begrepps- eller processmodelleringar. Användarna beskriver ofta situationen som en svårbegriplig och abstrakt till dess man börjar tala om utformningsaspekter som är sådant som är konkret och som man kan relatera till. Oavsett systemutvecklingsmetod så är vår erfarenhet att man alltid kan påbörja prototypframtagandet tidigare.

- **Prototyping för att samla krav.** Kan man rent av utnyttja tidiga prototyping-tekniker för att på så sätt vaska fram de funktionella kraven på systemet. Om detta är möjligt att göra kan man vinna tid, därför att man tidigare kommer fram till konkreta lösningar som användarna kan förhålla sig till, samt att man kan landa vid produkter som har betydligt större grad av användbarhet.
- **Deltagande prototyping.** Det kommer allt oftare rapporter om att användarna själva kan konstruera ganska bra verktyg för att stödja deras arbete, tvärtom vad vi tidigare brukar säga. Vore det därför inte tänkbart att involvera användarna i själva utformningsarbetet, dvs. att de deltar i, eller kanske rent av leder utformningsarbetet. Vi vet att användarna inte är några erfarna designers, men kan man få till stånd bra fungerande lag som kan samverka runt dessa aspekter, så kan deras respektive kompetens på ett mycket bra sätt tas tillvara på i utformningsarbetet.
- **Kontextuell prototyping.** Om vi omorganiserar på ett sådant sätt att vi kan placera ut utvecklarna direkt i verksamheten, mitt ibland användarna, så borde vi kunna uppnå spännande resultat. Den kommunikation som utvecklare sinsemellan behöver ha borde enklare kunna ske elektroniskt än den kommunikation som behöver ske mellan användarna eller mellan användare och utvecklare. Många goda idéer uppstår ofta över en kopp kaffe och med en direktkontakt mellan utvecklarna och användarna borde man underlätta för utvecklarna att faktiskt utnyttja användbarhetskompetensen, och att användarna faktiskt får betydligt mer inflytande över utvecklingen. Det torde dessutom vara svårare att utveckla svåra system när man lärt känna de personer som faktiskt skall använda det hela.

4. Användbarhetsdesignern

Att etablera en ny roll som användbarhetsdesigner är mycket viktigt för att säkerställa att användbarhet kommer in i utvecklingsarbetet. En sådan kompetens skall arbeta som länk mellan användarna och utvecklarna och behöver därför ha kunskap och kompetens inom områden såsom:

- Människa-datorinteraktion
- Viss konstnärlig designkunskap
- Förmåga att förstå den ofta komplexa arbetsdomänen
- Kunskap om systemutveckling

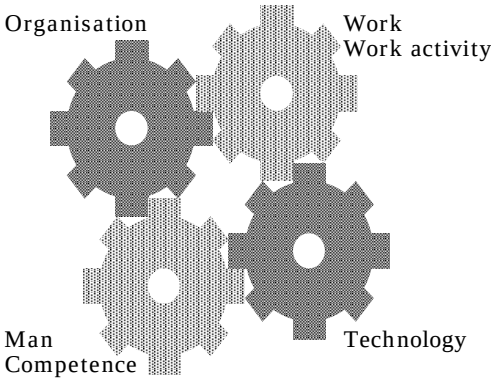
Det är mycket viktigt att denna roll inte förfaller till att bli en gränssnittsutvecklare. Mer om hur denna kompetens ser ut och hur den personen bör arbeta beskrivs i ett separat avsnitt (se avsnitt *8.2 Användbarhetsdesignern*).

5. Holistisk syn på utvecklingsarbete

Det är viktigt att förstå att IT-utveckling inte kan ske isolerat från övrigt utvecklingsarbete. Vi har varit med om situationer i vilka vi har undrat vilka de konsulterna som är med i projektet som står där borta är?

– Bry er inte om dem, de är organisationsutvecklarna!

Det är mycket viktigt att man med en helhetssyn kan beakta verksamhetsutveckling, IT-utveckling, organisationsförändringar, arbetsmiljöutveckling och kompetensutveckling. Alla dessa aspekter är beroende av varandra. (se figur nedan).



Figur 23. Helhetssyn på verksamheten.

6.4 Framtida arbetsformer och visionsseminarier

I samband med projekten SKATTEKONTO/MAGI på RSV var CMD inblandade i diskussioner runt framtida arbetsformer. En gemensam arbetsgrupp för framtida arbetsformer kallad »Arbetsformer« bildades gemensamt av de olika projekten. Denna arbetsgrupp fick inga givna direktiv utan fick själva definiera sitt ansvar. Detta såg man huvudsakligen som ett arbete där man skulle marknadsföra det nya arbetssättet som skulle behöva komma som en följd av de nya systemen som utvecklades inom 98-projekten.

Vad man snarare borde ha gjort var att definiera de framtida arbetsformerna och därefter bestämt hur systemen skulle ha sett ut för att stödja denna typ av arbete. Detta tyckte alla i projektet var fullkomligt rimligt och korrekt, och trots att det fortfarande fanns tid att ändra projektets arbetssätt och ta fram de framtida arbetsformerna vidtogs inga åtgärder.

Låt en grundlig genomgång av framtida arbetsformer i termer av scenarion och tillhörande prototyper utgöra grunden för besluten om systemutvecklingsprojektens karaktär, storlek och delleveranser (se avsnitt, *4.4 Verksamhetsutveckling kontra systemutveckling*).

7

Kommersiella modeller för systemutveckling

Begreppen modell, metod och process används mycket slarvigt när man talar i termer av hur systemutveckling bedrivs. Även de på den kommersiella marknaden etablerade produkterna rör ihop begreppen. Vad vi dock bör skilja tydligt på är att standarden ISO 13407 inte är en färdig systemutvecklings-{modell, process, metod}. Den kan betecknas som ett *koncept* och måste omsättas i systemutvecklings-{modeller, processer, metoder}.

I detta kapitel tänkte vi titta lite på kommersiella systemutvecklingsmodeller. Med kommersiella menar vi att de finns som någon slags produkt, tillgängliga för vem som helst att köpa och använda sig av. Det kan vara både i form av böcker (handböcker) och programvara. Det finns även modeller som används internt inom organisationer och som inte är allmänt tillgängliga, vi nämner mycket kort några av dessa mot slutet av kapitlet.

Arbetet med att titta på kommersiella modeller i förhållande till användbarhet och användarcentrering är ett arbete som vi bara har påbörjat och följaktligen inte är färdiga med. Här kommer vi att jobba vidare, och exempel på det fortsatta arbetet finns i avsnittet *7.6 Fortsatt arbete*.

7.1 Rational Unified Process

Rational Unified Process – RUP kan anses vara en etablerad kommersiell systemutvecklingsmodell. Den bygger på ett objekt-orienterat och ingenjörsmässigt angreppssätt och har ett starkt fokus på systemarkitektur – »An architecture-centric process« [Kruchten, 1998, kapitel 5]. RUP tar avstamp i vad som kallas »best practices«. Detta är »goda vanor« som utvecklarna/författarna av RUP skaffat sig under decenniers erfarenhet av objekt-orienterad systemutveckling [Kruchten, 1998, sidan 6]:

1. Develop software iteratively.
2. Manage requirements.
3. Use component-based architectures.
4. Visually model software.
5. Verify software quality.
6. Control changes to software.

Rational Unified Process – RUP tar alltså avstånd ifrån sättet att bedriva systemutveckling som ett enda stort vattenfall. Man poängterar dock förtjänster i ett strukturerat och definierat arbetssätt och föreslår en kontrollerad iterativ utveckling, där det stora vattenfallet är omgjort till en rad mindre och väl definierade vattenfall – *evolutionary prototyping*.

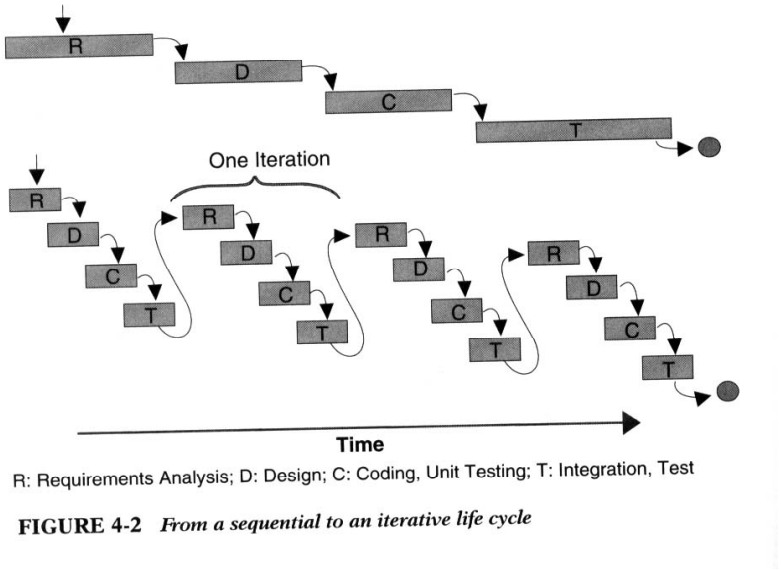
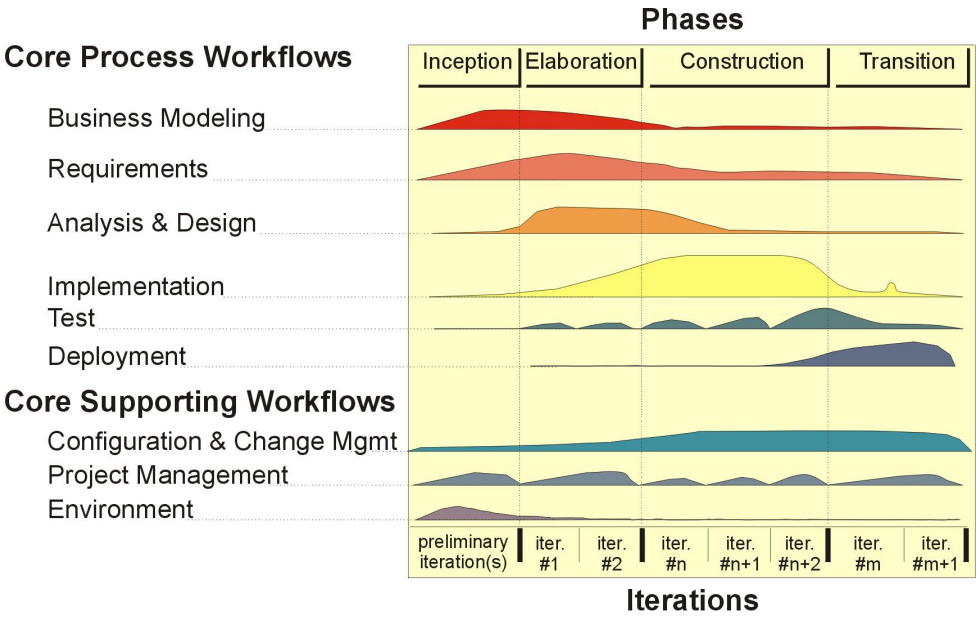


FIGURE 4-2 *From a sequential to an iterative life cycle*

Figur 24. Varje iteration är ett litet vattenfall.

RUP är uppdelad i ett antal faser med arbetsflöden/aktiviteter och iterationer:



Figur 25. Faserna och huvudprocesserna i RUP.

De finns fyra faser i processen: inception (uppstart), elaboration (utformning), construction (konstruktion) och transition (leverans). Dessa faser är tidsorienterade och innehåller var och en ett antal iterationer. Arbetet inom varje fas bedrivs i ett antal arbetsflöden eller aktiviteter. Betoningen av dessa aktiviteter skiftar under processens gång. I figuren ovan ser man t ex att business modelling har sin tyngdpunkt under faserna inception och elaboration.

Use-cases eller användningsfall är mycket centralt inom RUP. Ett användningsfall är en sekvens av transaktioner som ett system utför som respons på stimuli från en aktör. Vi tänker inte här redogöra för exakt vad ett användningsfall innebär och består av utan hänvisar till litteraturen, t. ex. Kruchten, 1998. Detta gäller även för modellbeskrivningsspråket UML (Unified Modeling Language) vilket har stort inflytande på RUP.

RUP OCH PROTOTYPING. Det är intressant och studera vad de olika utvecklingsmodellerna säger om prototyping. Detta för att man ganska tydligt kan se vad respektive modells tyngdpunkt ligger. RUP definierar fyra stycken olika prototyper:

- *Behavioral prototypes:* prova ett speciellt uppförande.
- *Structural prototypes:* utforska en systemarkitektur/teknik.
- *Exploratory prototypes:* utforska och kasta bort.
- *Evolutionary prototypes:* gradvis utveckling till produkten.

Vidare säger man att den viktigaste prototypen som görs är den som provar systemarkitekturen (structural prototype). Detta för att kunna verifiera att man håller på att ta fram en realiserbar lösning. Prototyper för användargränssnittet ses i första hand som en aktivitet i arbetsflödet requirements, och syftar till att hitta krav.

Man poängterar att utvecklingen av systemet (eller produkten) skall bedrivas efter principen om evolutionary prototyping. Övriga prototyper är sådana som används under kortare tid under utvecklingen.

Det är viktigt att inse att RUP inte innehåller något uttalat stöd för att bedriva iterativ *användarcentrerad* systemutveckling. Det som växer fram under utvecklingens gång är systemets arkitektur och de olika klasser och objekt som är grunden i det objekt-orienterade angreppssättet. Användbarheten, användargränssnittet och användarnas inverkan på systemets funktionalitet och utformning hanteras i tidiga faser av utvecklingen, för att sedan anses vara avklarade. Det handlar i RUPs fall framför allt om att tidigt hitta användarkraven. Detta innebär att det iterativa arbetssättet enligt ISO13407 med aktiviteterna *analys–design–utvärdering* inte följer med under hela systemutvecklingsprocessen. Fokus på användarna och deras arbetsuppgifter flyttas tidigt över till ett fokus på själva systemlösningen (arkitekturen).

7.2 Dynamic Systems Development Method

Dynamic Systems Development Method – DSDM är en modell (trots att metod nämns i namnet dristar vi oss till att benämna den modell) som vuxit fram

framför allt i England. Den är framtagen inom ramen för ett konsortium där många företag och enskilda »ställt upp« och bidragit med sina kunskaper och erfarenhet. Det syns en tydlig trend till att organisationer och företag visar intresse och tar till sig DSDM. Det kan därför finnas visst intresse att se vad DSDM har att erbjuda i jämförelse med RUP.

DSDM har nio principer som utgör grunden till modellen:

1. Active user involvement is imperative.
2. DSDM teams must be empowered to make decisions.
3. The focus is on frequent delivery of products.
4. Fitness for business purpose is the essential criterion for acceptance of deliverables.
5. Iterative and incremental development is necessary to converge on an accurate business solution.
6. All changes during development are reversible.
7. Requirements are baselined at a high level.
8. Testing is integrated throughout the life-cycle.
9. A collaborative and co-operative approach between all stakeholders is essential.

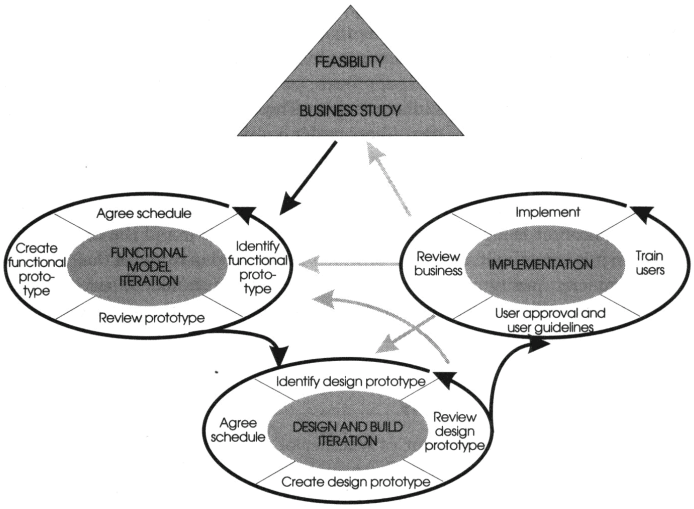


Figure 1.1 DSDM process diagram.

Figur 26. Faser och huvudprocesser i DSDM.

De fem faserna i modellen är:

- feasibility study
- business study

- functional model iteration
- design and build iteration
- implementation

DSDM bygger mycket på användarmedverkan och man har också definierat ett antal användarroller i projekten. Detta är i och för sig vettigt, men skall ses mot bakgrund av att man har inte definierat något sätt att få ett representativt urval av användare på, utan förespråkar ett urval där man fokuserar på ett effektivt arbete, framför allt i form av workshops.

Stora delar av arbetet inom ett DSDM projekt bygger på s. k. JAD-workshops (eng. *Joint Application Development*). Dessa workshops är mycket centrala inom DSDM och ses som det viktigaste metodinslaget. Enlig DSDM kan man applicera dessa JAD-workshops på nästan vilket problemområde som helst. Man ser workshop som ett effektivt redskap där man på kort tid snabbt kan komma fram till resultat och beslut. Det finns dock uppenbara risker att dessa JAD-workshops blir den enda metod som används. Här ser vi luckor vad gäller mer analytiska metoder och strukturerade arbetssätt bl. a. i arbetet att analysera användare, arbetsuppgifter och sätta upp användbarhetsmål, samt att genomföra formella (och mer informella) utvärderingar mot uppställda mål. Det kan lätt bli så att de representanter för användarna som deltar i dessa workshops inte längre har en förankring i den ordinarie verksamheten etc. Man missar också en mycket viktig bas för informationsinsamling och analys då man väljer att enbart ta in vissa definierade användarrepresentanter och jobba med dem i workshops, och inte gå »ut i fält« och studera, analysera etc.

DSDM OCH PROTOTYPING. Inom DSDM förespåkar man följande prototyper:

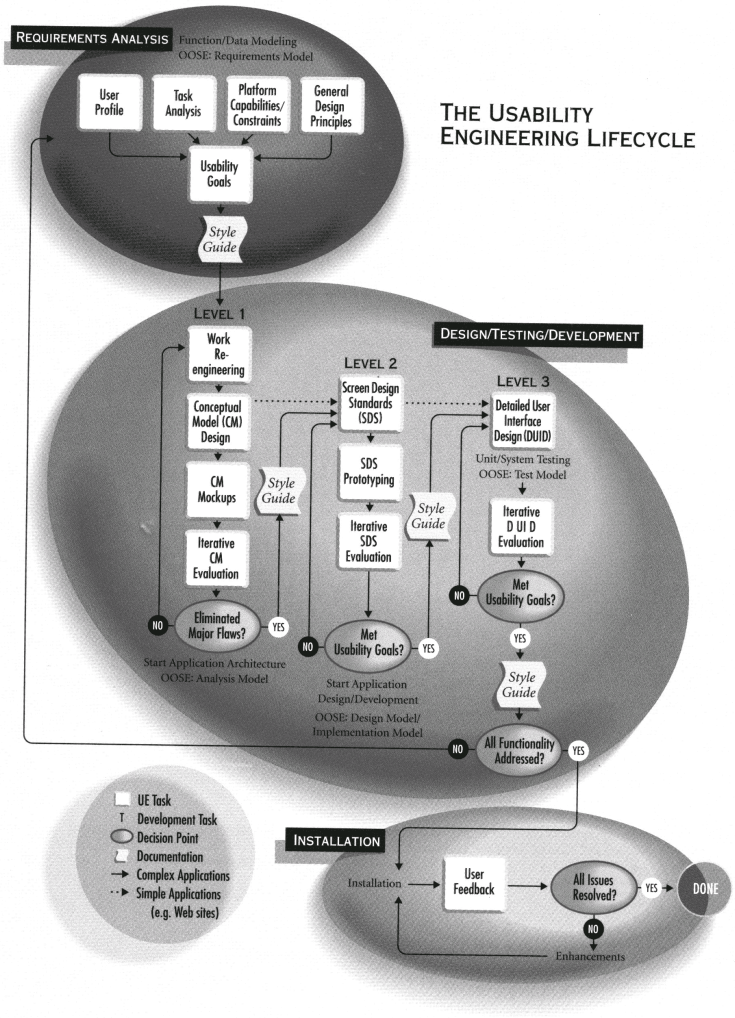
- *Business prototypes:* prova funktionalitet
- *Usability prototypes:* utforska användargränssnittet, som ej påverkar funktionaliteten
- *Performance and capacity prototypes:* försäkra sig om att systemet klarar belastning etc
- *Capability/design prototypes:* prova en designlösning

Utvecklandet av själva systemet kan i DSDM:s fall ses som inkrementell prototyping. Man fokuserar ganska mycket på prototyping och ser det som viktiga byggstenar i utvecklingsarbetet. Centralt är den hårda tidsstyrningen och den s.k. »time-boxing:en«. Man delar upp systemet i olika leveranser och ser till att hålla tiden, och i stället styr man med funktionaliteten.

DSDM är inte alls lika detaljerad och långt driven som RUP. Där RUP baseras på ett modelleringsspråk (UML) och till stor del även på verktyg (Rational Rose m. fl.) är DSDM mer ett ramverk att utgå ifrån. Dock pågår mycket arbete inom konsortiet att ta fram verktygsstöd etc.

7.3 The Usability Engineering Lifecycle

Vi vill också visa på en systemutvecklingsmodell som tar »fullt« ansvar för användbarheten. Deborah J. Mayhew försöker in sin modell, beskriven i boken »The Usability Engineering Lifecycle« [Mayhew, 1999] ta ett helhetsgrepp om användbarheten:



Figur 27. Huvudprocesserna i »The usability engineering lifecycle«.

Hon är noga med att påpeka att hennes modell avser att komplettera t. ex. en befintlig systemutvecklingsmodell med nödvändiga delar för att säkerställa

användbarheten i de system som skall utvecklas. Man kan t. ex. tänka sig »mixa« RUP eller DSDM med denna modell.

Mayhews har en intressant vinkling där hon pratar om en Product Style Guide. För RSVs del kan det vara intressant att se den nivå hon definierar en Product Style Guide på i jämförelse med andra typer av Style Guides [Olsson, 1999]. Hon anser att man skall plocka in bl. a. följande delar i en Product Style Guide för en produkt:

- Overview of Product Functionality.
- User Profiles.
- Contextual Task Analysis.
- Platform Capabilities and Constraints.
- Usability goals.
- Reengineered Work Models.
- Conceptual Model Design.
- Input and Output Devices.
- Screen Design Standards.
- Feedback.

Vi ser att en sådan Product Style Guide innehåller fler specifika användbarhetsrelaterade aspekter än vad en domän- eller företagsspecifik Style Guide gör. Dessa har mer en vinkling mot ett gemensamt utseende och uppförande. Det vore intressant att försöka ta fram en Product Style Guide inom något projekt, exempelvis parallellt med den Style Guide som redan finns inom RSV.

7.4 Några andra modeller

Som vi nämnde i inledningen av detta kapitel finns det rad andra modeller som är mer eller mindre kommersiella men företags- eller organisationsinterna:

- Cap Gemini har en modell som de kallar Iterative Application Development. Det är en vidareutveckling av en tidigare och enklare Rapid Application Development modell.
- Praktisk Användarmedverkan vid Systemutvecklingen är framtagen av Enator.
- DELTA-metoden är utvecklad i samarbete mellan Ericsson Infocom och Linköpings universitet. Den används bl. a. av Ericsson och WM-data.

7.5 Kommersiella modellers förhållande till användbarhet och användarcentrering

Vad gäller användbarheten och användarcentrering inom de olika modellerna vill vi i detta läge att bara göra några kommentarer kring de två just nu mest uppmärksammade: RUP och DSDM. Här behöver vi, och kommer vi, att fortsätta arbetet för att se huruvida man behöver lägga till/förändra delar av dessa modeller för att bli »fullt« användarcentrerade (se *7.6 Fortsatt arbete* nedan).

RUP. Har sin styrka i det kontrollerade och iterativa arbetssättet. Dock ser vi en hel del svagheter vad gäller användbarhetsaspekter och användarcentreringen. Mycket kort rör detta bl. a.:

- Användbarhet är inte explicit uttryckt i modellen.
- Användbarhet tas upp i termer av gränssnittsdesign och inte i sin hela bredd.
- Användbarhet ses som ett icke funktionellt krav. Detta leder bl. a. till användbarheten inte uttrycks i mätbara termer.
- Några explicita tester av användbarhet uttrycks inte.
- Användarmedverkan är mycket vagt uttryckt.
- Den projektroll som har en anknytning till användbarhet benämns User Interface Designer och ses just som en gränssnittsdesigner och inte en person med användbarhet som kompetensområde.
- Analyser av användarnas arbetsuppgifter och användargrupper uttrycks inte i modellen.
- Målformuleringar rörande användarnas arbetsuppgifter finns inte med.
- Det finns uppenbara risker att användarna och användbarhet bara finns med i början av projekt för att sedan försvinna mer och mer allt eftersom systemets arkitektur sätts i fokus.

DSDM. Vad gäller DSDM ser vi vissa svagheter i den teoretiska bakgrunden samt det praktiska genomförandet. DSDM bygger på erfarenheter från en rad projekt och företag, men har ingen »modern« förankring vad gäller användbarhet och användarcentrering. Man har inte insett eller förstått den fulla bredden av användbarhet och ett användarcentrerat arbetssätt. Det finns en väldig fokusering på tidsaspekten (time-boxing:en) och JAD-worlshops. Här ser vi en risk att många aspekter på användbarhet missas. Vidare ser vi tveksamheter kring de användarroller som finns, eller snarare de som inte finns. Vi tror att det lätt kan bli så att de i projektet definierade rollerna anses kunna

leverera svaret på alla frågor kring användarnas krav, behov etc, oavsett hur användarkategorierna ser ut i målverksamheten.

Generellt för båda modellerna är att de inte har någon klar förankring i ISOs definition av användbarhet och användarcentrerad systemutveckling. Vi ser att det saknas både aktiviteter (processer och metoder) och roller inom modellerna som säkerställer utvecklandet av en användbar produkt.

7.6 Fortsatt arbete

De systemutvecklingsmodeller och ramverk som har beskrivits i föregående sektion är ingenjörsmässiga modeller över hur processer skall gå till så till vida att de beskriver aktiviteter i boxar med pilar emellan. Designprocessen är ofta betydligt mer ostrukturerad till sin natur. Frågan är om den skall vara det eller om den borde inordnas i ett mer ingenjörsmässigt ramverk? Detta är till syvende och sist ett pedagogiskt problem, hur kan vi lära systemutvecklare och övriga projektdeltagare ett användarcentrerat angreppssätt? Måste vi för att förklara vad användarcentrerad systemutveckling innebär stoppa in de olika aktiviteterna i boxar för att man initialt sett skall förändra arbetssättet vid utveckling, eller kan man på något annat sätt förändra organisationens kunskaper och attityder.

Det är idag mycket vanligt att man, i det att man ser över sin systemutvecklingsverksamhet i syfte att förbättra och effektivisera, bedömer det som bästa att köpa in en ny systemutvecklingsmodell. Detta innebär givetvis många positiva saker, dagens systemutvecklingsmodeller är ofta mycket mera heltäckande och integrerade med verktyg. Det innebär också ofta att man inte tar till vara på den kompetens om befintliga metoder och rutiner som finns inom organisationen, man brukar inte heller kunna behålla vad som är bra med befintliga modeller.

Att införskaffa nya systemutvecklingsmodeller, som t. ex. RUP, kan innebära ett steg tillbaka i fråga om användbarhet och användarcentrerad systemutveckling. De flesta av de nya modellerna har brister vad avser dessa aspekter och dessa brister är dåligt dokumenterade.

Vi avser fortsättningsvis att studera flera kommersiella systemutvecklingsmodeller med avseende på just användbarhet och användarcentrerad systemutveckling. Här följer en kort beskrivning av den arbetsmetod vi planerat att använda.

1. Ta fram en aspektlista med de aspekter som vi anser som särskilt viktiga för att systemutvecklingen skall inbegripa användarna och fokusera på användbarhet.

2. Gå igenom befintliga modeller i termer av de teoretiska beskrivningarna och dokumentationen för att kunna bedöma hur pass väl aspekterna i aspektlistan stöds av modellen.
3. Intervjua personer som använt modellerna både:
   - projektledare utan någon specifik kunskap om/erfarenhet av användbarhet och
   - användbarhetsexperter i projekten.
4. Försöka att för varje modell beskriva vad de behöver kompletteras med för att mera kunna fokusera på användbarheten och användarcentrerad systemutveckling.
5. Vid behov specificera och tillämpa de kompletterande metodsteg/specifikationer som systemutvecklingsmodellerna behöver kompletteras med.
6. Utvärdera detta i verkliga projekt, gärna då i de organisationer som vi tidigare intervjuat (se punkt 3).

8

Vissa viktiga roller

Begreppet användare är mångtydigt. Vem är egentligen användare av en tjänst? Den som handlägger ett ärende? Chefen? Eller kanske rent av kunden som använder en elektronisk tjänst. Definitionen av begreppet användare är problematisk, det är inte alltid alldeles tydligt vilka individer som ryms bakom begreppet. Konkreta kontakter mellan användare och projekt kan vara med köparen av ett nytt system, cheferna tillpersonerna som faktiskt är de som i sluttampen skall få använda systemet. Hur man skall samverka med användarna beror till väldigt stor del på vilka användarna i själva verket är. Notera att en chef eller en person i ledande ställning ofta tror och hävdar att denne är en typisk, representativ användare, men ofta har denne ingen aning om hur hans »underlydande« i själva verket arbetar, vilket brukar vara ganska långt från sanningen. Användare som lånas in i projekt för längre tid förvandlas snabbt från att vara en representativ, »typisk« användare till att vara mer en typisk professionell projektmedlem, de förändras från att vara en förespråkare av användarnas behov till att bli en försvarare av projektet.

8.1 Att använda användare – riktlinjer för användarmedverkan

Vi har tidigare konstaterat att användarcentrerad utveckling kräver »effektiv användarmedverkan«. Med effektiv användarmedverkan menar vi att det ska

vara rätt användare vid rätt tillfälle. I avsnitt *6.3 Hur skulle RSVs systemutvecklingsmodell kunna utvecklas*, tog vi upp en del aspekter kring urval av användare och användarmedverkan. Vi kommer nu att fördjupa oss ytterligare i problematiken samt, återigen, använda RSV som ett praktikfall. Vi anser att de principer som redogörs för i detta avsnitt bör gå att applicera inom andra organisationer med liknande uppbyggnad.

Under de senaste fyra till fem åren har det inom RSV-koncernen bedrivits olika projekt/utvecklingsinsatser med olika förhållningssätt till användarmedverkan. I ett fall har man bedrivit utvecklingsarbetet på tre kontor med ett fåtal engagerade handläggare på respektive kontor. I ett annat fall har man bedrivit en utvecklingsinsats med i stort sett samtliga användare (positiva och negativa) på ett kontor. Referensgrupper för de sk 98-projekten har utsetts enligt olika principer. Grupperna har bestått av användare, som lämnat synpunkter på olika rapporter m.m., har valts ut på olika sätt. I ett fall har samtliga representanter kommit från Stockholmsområdet. I det andra fall har de kommit från olika delar av landet. För att få rätt användare krävs särskilda urvalsmetoder. Först måste man emellertid fastställa de egenskaper och faktorer som bör finnas representerade i utvecklingsarbetet.

Vid utvecklingsarbete ska:

- Hela verksamheten vara representerad såväl den praktiska som den materiella kompetensen (med materiell kompetens anses framför allt kunskap om lagstiftning och regelverk inom skatteväsendet). Det är önskvärt att ha flera personer med samma kompetens för att kunna utgöra diskussionspartners/stöd för varandra.
- Helhetssynen vara representerad, dvs förmågan att se vilka konsekvenser verksamhetsutvecklingsinsatsen har på personal, organisation och teknik.

8.1.1 Användare och verksamhetsexperter

I huvudsak kan man se två tydliga grupperingar av användare; slutanvändare och verksamhetsexperter. I RSVs fall definieras dessa enligt nedan:

SLUTANVÄNDARE är en person, som skall använda det IT-stöd som utvecklas eller köpts in. Ett projekt kan ha behov av slutanvändare i två olika roller:

- **Roll 1** deltar i samtliga de olika modelleringar som utförs under projekttiden. Projektet ska tillgodogöra sig hennes kunskaper och det finns inga krav på att hon ska dokumentera o dyl. Svarar för kontinuiteten av användarsynpunkter/-medverkan och operativ verksamhetskunskap.

Kompetenskrav: god kunskap om den verksamhet som utvecklingsinsatsen ska stödja. Ska vara en »superanvändare« med god samarbetsförmåga: utvecklingsarbetet innebär att man medverkar i en kreativ process tillsammans med andra människor. Ska vara lyhörd och samtidigt kunna hävda sina ståndpunkter helhetsförståelse. Det är viktigt för helheten att alla synpunkter diskuteras. Det är också viktigt för helheten att den enskilde projektdeltagaren inser att hans synpunkter är viktiga, men att de ibland måste stå tillbaka för andra mer viktiga. Ha intresse för utvecklingsarbete och då speciellt inom det aktuella området. Vara förändringsbenägen och öppen får nya infallsvinklar

- **Roll 2** ska lämna synpunkter på olika produkter som arbetats fram i projektet. En aktivitet är t ex att testa prototyper av olika slag. Kan vara olika personer som innehar Roll 2 vid olika tillfällen under utvecklingsarbetet. Det finns inga krav på kontinuitet. Ska utgöra ett representativt urval av användare.

  Kompetenskrav: bör ha grundläggande kunskap om den verksamhet som utvecklingsinsatsen ska stödja ska vara en »vanlig« användare.

VERKSAMHETSEXPERT–MYNDIGHET ska täcka verksamheten med »praktisk och materiell« kompetens. Hon ska aktivt bidra till att utvecklingsarbetet fortskrider. Hon lånas normalt in till RSV på heltid under en begränsad tidsperiod. Initialt är hon slutanvändare för att efter en tid (ca en till två månader) transformeras till Verksamhetsexpert-myndighet. Efter ytterligare en tid transformeras hon till att bli Verksamhetsexpert-RSV (se nedan).

Kompetenskrav: god kunskap om den verksamhet som utvecklingsinsatsen ska stödja. God samarbetsförmåga: utvecklingsarbetet innebär att man medverkar i en kreativ process tillsammans med andra människor. Ska vara lyhörd och samtidigt kunna hävda sina ståndpunkter. Helhetsförståelse: det är viktigt för helheten att alla synpunkter diskuteras. Det är också viktigt för helheten att den enskilde projektdeltagaren inser att hans synpunkter är viktiga men att de ibland måste stå tillbaka för andra mer viktiga. Det krävs även att han har förmåga att se vilken påverkan utvecklingsarbetet har på andra verksamheter/verksamhetsgrenar. Ha intresse för utvecklingsarbete och då speciellt inom det aktuella området. Vara förändringsbenägen och öppen får nya infallsvinklar önskvärt med organisatorisk kunskap. Van vid att utreda och dokumentera resultatet av utredningen.

VERKSAMHETSEXPERT-RSV ska täcka verksamheten med »praktisk och materiell« kompetens. Hon ska aktivt bidra till att utvecklingsarbetet fortskrider.

Hon arbetar normalt med administrativ utveckling inom RSV (förvaltningsansvarig eller liknande).

Kompetenskrav: god kunskap om den verksamhet som utvecklingsinsatsen ska stödja god samarbetsförmåga: utvecklingsarbetet innebär att man medverkar i en kreativ process tillsammans med andra människor. Ska vara lyhörd och samtidigt kunna hävda sina ståndpunkter. Helhetsförståelse: det är viktigt för helheten att alla synpunkter diskuteras. Det är också viktigt för helheten att den enskilde projektdeltagaren inser att hans synpunkter är viktiga men att de ibland måste stå tillbaka för andra mer viktiga. Det krävs även att han har förmåga att se vilken påverkan utvecklingsarbetet har på andra verksamheter/verksamhetsgrenar. Helhetssyn, dvs förmåga att se vilka konsekvenser verksamhetsutvecklingsinsatsen har på personal, organisation och teknik. Ha intresse för utvecklingsarbete och då speciellt inom det aktuella området. Vara förändringsbenägen och öppen får nya infallsvinklar van vid att utreda och dokumentera resultatet av utredningen.

8.1.2 Urvalsfaktorer

I olika sammanhang har man konstaterat att blandade grupper ökar kreativiteten och möjligheten till ett bra och allsidigt resultat. Det finns vissa tecken på att det bästa utvecklingsresultatet uppnås då man »maximerar skillnaden« i en arbetsgrupp. Med detta menas att man i en grupp t ex har en nyanställd och en som arbetet lång tid i verksamheten. För att få en rätt sammansättning av en arbetsgrupp bör följande faktorer beaktas:

- **Kvinnor och män**. Det är alltid en fördel om det finns både manlig och kvinnlig representation i en arbetsgrupp. Då man utvecklar användargränssnitt är det viktigt med en ordentlig kvinnlig medverkan eftersom kvinnor ofta är mer inriktade på mänskliga behov än på datorernas tekniska prestanda.
- **Ålder.** Denna kan spegla åldersfördelningen för dem som arbetar i aktuell verksamhet. Det kan emellertid vara betydelsefullt att alla åldrar är representerade. Som ovan sagts kan det finnas fördelar med blandade grupper och att man »maximerar skillnaden«, dvs. unga och gamla tillsammans. Detta är speciellt viktigt vid testning av prototyper och motsvarande.
- **Erfarenhet.** I utvecklingsarbetet bör det finnas både experter på verksamheten och personer som är relativt nya i verksamheten.
- **Datorvana.** För att kunna utveckla ett användbart IT-stöd för alla bör inte utvecklingsgruppen bestå enbart av datorvana och tekniskt

intresserade personer. Det är betydelsefullt att sådana personer finns med, men det är kanske än viktigare att det finns någon som representerar de som inte är så avancerade i sin datoranvändning. I dagens läge är dessa troligtvis i majoritet och bör därför också vara i majoritet vid testning av prototyper och motsvarande.

- **Användarkategori.** Samtliga de kategorier som ska utnyttja IT-stödet bör vara representerade. Kategorier som kan vara aktuella är t.ex. chefer, experter, »vanliga« handläggare, kvalificerade assistenter och »vanliga« assistenter.
- **Storlek** (antal anställda) på det kontor som slutanvändaren och verksamhetsexpert-myndighet kommer ifrån. Olika storlekar bör vara representerade. Det är viktigt att bl.a. beakta de speciella förhållanden som finns i en storstadsregion respektive i en glesbygdsregion.
- **Antal och typ** av ärenden som handläggs på det kontor slutanvändaren och verksamhetsexpert-myndighet kommer ifrån. Ska resultatet av utvecklingsarbetet t.ex. stödja företagsgranskningen bör i första hand representanter tas från kontor som har företag inom sin domän.
- **Geografisk spridning.** Det är inte lämpligt att endast en geografisk region är representerad i utvecklingsarbetet. För att uppnå ett användbart IT-stöd bör deltagarna i utvecklingsarbetet komma från olika delar av riket.
- **Gruppstorlek.** Ett projekt kan ha relativt många deltagare. I många fall är det lämpligt att dela in projektet i olika arbetsgrupper, som var och en får sin specifika uppgift. Vid vissa aktiviteter som t.ex. datamodellering bör gruppen enligt erfarenhet dock helst bestå av 6 - 8 personer och maximalt till 10 –12 personer.

De ovan nämnda faktorerna är i vissa fall motstridiga och går inte att uppfylla samtidigt. Det gäller som vid så många andra tillfällen att göra en samlad bedömning. Då måste man även väga *nyttan* mot *kostnaden*.

8.1.3 Urvalsmetoder

Det är i allas intresse att utvecklingsprojekten bemannas med »rätt« deltagare. Utvecklingsinsatsen ska nämligen leda till ett IT-stöd, som oftast ska användas av många. Projektledning (uppdragsgivare, styrgrupp och projektledare) har naturligtvis ett väsentligt intresse av att det är »rätt« deltagare. Även övriga berörda såsom chefer på olika nivåer i regionerna samt de som ska använda det kommande IT-stödet är betjänta av att det är »rätt« deltagare.

Vid valet av deltagare bör det därför alltid göras en lämplighetsprövning. Detta är särskilt viktigt vid valet av verksamhetsexperterna (myndighet och RSV) och de slutanvändare (Roll 1) , som ska delta i modelleringar m.m. Genom att göra intervjuer och använda olika »testverktyg« kan man utröna om en person har de önskvärda egenskaperna för att få delta i utvecklingsarbete. Det finns »testverktyg« som går att använda för att få fram vilken grupparbetsroll en viss person tillhör. Ett verktyg, som innehåller drygt 60 frågor, delar in oss i åtta olika grupparbetsroller: praktikern, innovatören, entreprenören, samspelsingenjören, utvärderaren, ordföranden, genomföraren och resursforskaren. Oftast visar resultatet att man har flera roller varav av en brukar vara dominerande. Varje roll beskrivs med ett antal egenskaper. Exempelvis för **praktikern** anges sålunda:

- Omsätter beslut i handling.
- Disciplinerad, systematisk.
- Föredrar beprövade metoder och stabila strukturer.
- Är konkret och verklighetsnära.
- Ogillar förändringar.
- Svårt att koppla framtidsperspektiv.

Innovatören beskrivs på följande sätt:

- Idésprutan, planterar nya idéer.
- Tilltalad av de stora perspektiven.
- Utmärks av originalitet.
- Fångas hängivet av sina idéer.
- Ointresserad av genomförandet.
- Ofta orealistisk, vill inte se de praktiska begränsningarna.

I RSV-koncernens »verktygslåda« för att välja ut deltagare till utvecklingsarbete bör det finnas ett urvalsverktyg med ungefär det innehåll som beskrivits ovan.

Det finns olika sätt att rekrytera deltagare till utvecklingsarbete. Nedan beskriver vi några olika sätt. Utvecklingsarbetets art och omfattning påverkar naturligtvis valet.

- **Annonsering.** Inbjudan till koncernens personal att anmäla intresse att medverka i redan planerat/pågående utvecklingsarbete. Detta kan ske i Notes eller på annat lämpligt sätt. För att ha en framtida beredskap kan man även göra en inbjudan till koncernens personal att

anmäla intresse att medverka i utvecklingsarbete inom visst verksamhetsområde. På detta sätt kan koncernen bygga upp en »bank« av personer, som är intresserade av utvecklingsarbete.

- **Handplockning.** RSV frågar myndighets-/kontorschef om det går för sig att låna viss namngiven person alternativt om det finns någon lämplig person som kan delta i utvecklingsarbete.
- **Nominering.** Användare väljer någon som ska vara deras representant i utvecklingsarbetet.

Förfarandet för urval av projektdeltagare måste vara allmänt accepterat av verksledningen och regioncheferna. Det är viktigt att RSV alltid tar kontakt med den blivande projektdeltagarens chef(er). Detta bör ske tidigt i urvalsprocessen. Att medverka i utvecklingsarbete är väldigt olikt det arbete som man normalt utför på skattemyndigheten och kronofogdemyndigheten. RSV måste därför på ett tidigt stadium förklara vad det innebär för en person att medverka i utvecklingsarbete. Detta kan lämpligen göras redan i samband med annonseringen av inbjudan.

För det första bör RSV förklara vad deltagaren ska bidra med, t.ex.:

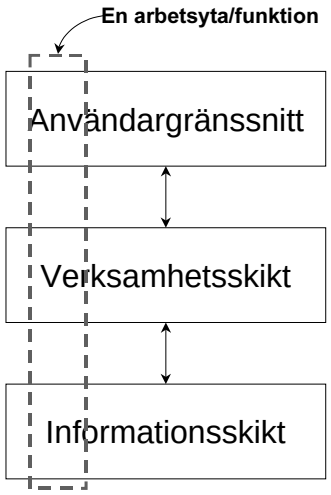
- Vilka kompetenskrav som ställs.
- Att hon ska ansvara för viss uppgift.
- Att hon självständigt ska göra utredningar och dokumentera dessa.
- Att hon har ett ansvar (gemensamt med de andra medverkande) för att utvecklingsarbetet fortskrider.

RSV bör också göra en kort presentation av utvecklingsmodellen samt klargöra under vilka förutsättningar deltagandet ska ske, t.ex.:

- Att arbetet ska bedrivas på viss ort.
- Att arbetet ska bedrivas under viss(a) tidsperiod(er).
- Att arbetet ska bedrivas på heltid, några dagar i veckan eller på annat sätt.
- Att deltagaren är en »kugge« bland flera i ett arbetsteam och att hennes insats därför är viktig för att utvecklingsarbetet ska få ett lyckat resultat.
- Att deltagaren representerar; endast sig själv, sitt verksamhetsområde, sitt kontor/region, samtliga användare.
- Att arbetet är förenat med visst lönetillägg (eller inte).

8.2 Användbarhetsdesignern

Våra studier visar att systemutvecklarna oftast ser sitt arbete som problemlösning. I mångt och mycket är ju systemutvecklingsarbetet förknippat med just detta. Effekten blir bl. a. att relativt mycket tid spenderas på att hitta eleganta programmeringslösningar och finna tekniklösningar. Vidare eftersträvas oftast att hitta lösningar för att konstruera en helhet; dvs ta ansvar för att utveckla en hel funktion från användargränssnitt till informationslagret, eller databasen. Detta innebär att exempelvis design av ett användargränssnitt blir en ganska liten del av utvecklarens hela arbetsuppgift:



Figur 28. En viss funktion genom de olika systemskikten.

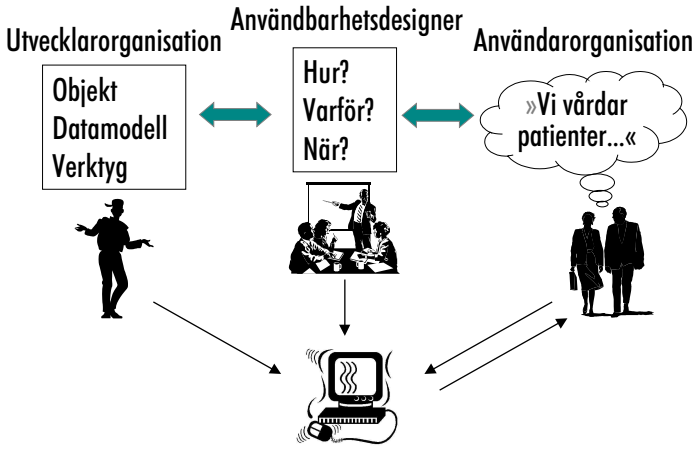
Utvecklarna ser helt naturligt den stora utmaningen i problemlösningen, och levererandet av ett slutgiltigt resultat; exempelvis delfunktionen genom alla tre lagren.

Det visar sig dock att detta inte alltid ger så bra resultat. Det blir svårt för utvecklarna att behärska alla kunskaps- och kompetensområden, samtidigt som tidsfaktorn gör att den mesta tiden läggs på att »få systemet att snurra«.

ANVÄNDBARHETSDESIGNER. Termen design, med syfte att beskriva utvecklandet av ett användargränssnittet, blandas tyvärr ofta lite slarvigt ihop med design som beteckning på hela utvecklingsprocessen, vilket leder till att alla systemutvecklare också automatiskt blir användargränssnittsdesigners. Detta är vare sig bra eller riktigt eftersom endast en ganska liten del av alla beslut som tas under utvecklingen är direkt relaterad till design av användargränssnittet. Design av användargränssnitt kräver speciell kunskap, kompetens och är inte bara sunt förnuft. Vi argumenterar starkt för att designarbetet skall utföras av experter på området. Den yrkesroll som skulle kunna ha som uppgift att säkerställa användbarheten och designa

användargränssnitten kallar vi – *användbarhetsdesigner* (A-designer). Denna roll har motsvarigheter i andra delar av systemutvecklingen, exempelvis av systemarkitekten (IT-arkitekt) som ansvarar för att systemets interna arkitektur blir effektiv och ändamålsenlig. Ytterligare en roll kan vara dataansvarig som har som uppgift att säkerställa att data via konceptuella modeller och design av databaser blir korrekt.

Det är A-designern som är ansvarig och använder sig av de användbarhetsrelaterade aktiviteter vi redogjort för i olika sammanhang i detta dokument. A-designern behöver kunskap och kompetenser inom områden såsom psykologi, beteendevetenskap, formgivning, typografi, systemvetenskap, datalogi och utvecklingsverktyg. Alla dessa områden ingår i forskningsområdet människa-datorinteraktion (MDI). Detta område och forskningsämne är det som på ett strukturerat och vetenskapligt sätt försöker skapa metoder och arbetssätt för att utveckla användbara system. Användbarhetsdesignern måste ha allmän metodkunskap och en djup kunskap och förståelse för MDI. Rollen blir en slags tolka eller länk, i en position mittemellan användarna (nyttjarna av systemet) och systemutvecklarna.



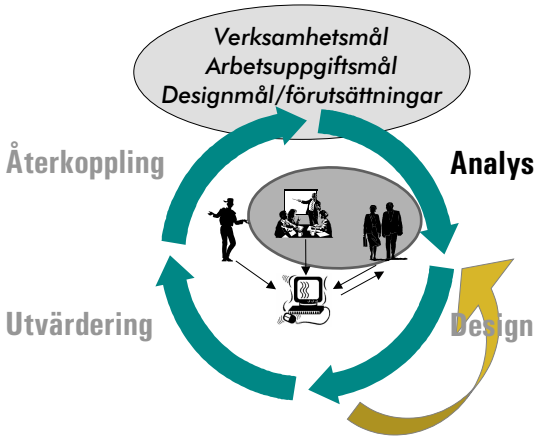
Figur 29. Användbarhetsdesignern som länken mellan utvecklar- och användarorganisationen.

(Slut-) användarna interagerar med systemet och skapar sig en bild av systemet. Det är den användbarhetsdesigners jobb att se till att denna bild stämmer överens med de arbetsuppgifter som användaren skall utföra med hjälp av systemet. Utan denna roll finns det en uppenbar risk att utvecklarorganisationen skapar ett system som inte speglar användarorganisationens bild utan snarare utvecklarnas uppfattning.

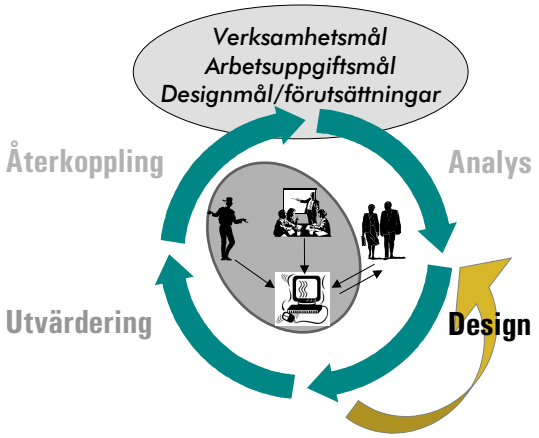
A-designern bedriver sitt arbete i nära samarbete med både användarna och utvecklarna. Rollens fokus flyttas helt naturligt under arbetets gång; då analysarbete övergår i mer av design och efterföljande utvärderingar samtidigt

som systemet växer fram från en skissartad prototyp till en färdig produkt. A-designern bör (för att inte säga skall) delta i alla utvecklingsfaser för att dels säkerställa att man arbetar användarcentrerat, och dels för att skaffa sig en bra bild av helheten. A-designern kommer därmed att fungera som en brygga över de gap som finns mellan de olika faserna.

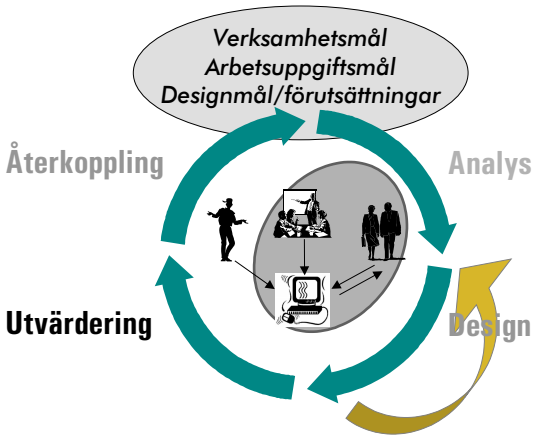
Analys – fokus på användarna och deras arbetsmiljö, arbetsuppgifter etc. Själva konstruktionen av systemet är inte i fokus.



Design – fokus flyttas delvis mot systemutvecklarna och systemets utveckling, samtidigt som användarna deltar i designaktiviteterna.



Utvärdering – användarna deltar/genomför utvärderingarna t ex genom användbarhetstester. A-designern deltar i utvärderingsaktiviteterna, men behöver nödvändigtvis inte genomföra dem.



Figur 30. Huvudfokus skiftar under det utvecklingens gång.

Notera att samspelet mellan A-designern, användarna och utvecklarna fortgår under hela den iterativa utvecklingsprocessen. Naturligtvis är det önskvärt att utvecklarna också har direktkontakt med användarna. Det finns ingenting i A-designerns roll som motsäger detta, det har dock visat sig att man mycket lätt tappar fokus på användbarheten och användarnas bästa, om man inte har en sådan tydlig ansvarsroll i projekten. A-designerns ansvarsområden kan sammanfattas som:

- Ser till att utvecklingsprocessen hålls användarcentrerad.
- Deltar aktivt i designprocessen och de designbeslut som rör användbarheten.
- Deltar i alla utvecklingsfaser och användbarhetsrelaterade aktiviteter.
- Jobbar nära både användarna och utvecklarna.
- Vara »specialist« inom området människa-datorinteraktion.

Se även beskrivning av användbarhetsdesignern i referensen *Göransson, Sandbäck, 1999*.

Man kan fråga sig om denna användbarhetsdesigner ersätter en hel organisation för användbarhet (sammansatt av många roller)? Det gör den naturligtvis inte, men den kan ses som en »billig« variant på delar av en sådan organisation. Rollen innehåller också viktiga inslag av kontinuitet som hur som helst är mycket svåra att upprätthålla i en större organisation. Vår bedömning är att det behövs A-designerns även i en mer komplett användbarhetsorganisation.

I större organisationer/projekt, eller när det finns större resurser, bör rollen kompletteras med andra permanenta roller i form av exempelvis grafiska designers och interaktionsdesigners.

8.3 Vikten av att gå ut i verksamheten

Användarmedverkan och användarnas deltagande i utvecklingsprojekt ses oftast som att plocka in användare, eller användarrepresentanter, i projektgruppen. Det är viktigt att göra detta, men det är lika viktigt att se till att utvecklarna kommer närmare användarna och deras arbetssituation genom att låta dem gå ut i verksamheten och studerar användarna. Det är kritiskt att de som har ansvar för användbarheten och designen av användargränssnittet verkligen går ut till användarna på deras arbetsplatser och studerar, intervjuar, etc. när de utför sina arbetsuppgifter. Det finns inget annat sätt att få reda på vilka arbetsuppgifter

som utförs och hur det utförs, annat än att vara med användarna när de utför dem.

Man kan gärna pröva olika typer av lösningar där man låter utvecklare bedriva analys- och prototypingarbete i samma fysiska lokaler som användarna jobbar i. Detta ger bl a möjligheter till informella och spontana kontakter.

9

Sammanfattning

Avslutningsvis tar vi här upp ett antal punkter vi har funnit viktiga när man som systemutvecklarorganisation skall försöka närma sig en användarcentrerad systemutvecklingsmodell:

- Introducera **användbarhetskompetens** i utvecklingsarbetet.
- Organisationen måste **själv specifisera sin användarcentrerade systemutvecklingsmodell**, gärna baserat på en kommersiell modell som t. ex. Rational Unified Process (RUP). Det är oerhört viktigt att systemutvecklingsmodellen känns som sin egen.
- Börja **arbetet med gränssnittet betydligt tidigare** under utvecklingen, tidigare faser behöver inte vara klara för att man skall inleda arbetet med gränssnittet.
- Använd prototyping-tekniker, och andra **användarcentrerade aktiviteter**, för att ta fram tidiga skisser av gränssnittet.
- **Kontextuell prototyping.** Överväg att placera utvecklarna ute på fältet mitt ibland användarna. Utvecklarna har mycket enklare att kommunicera med varandra på distans, däremot är motståndet stort

mot att försöka kontakta användarna om man sitter i en enhetlig utvecklarmiljö.

- **Kontinuerlig utveckling.** Arbeta för att projekten delas in i små hanterbara delar som kan utvecklas under en begränsad projekttid, säg tre månader. Arbeta för att alla system hela tiden utvecklas. Man blir aldrig färdig med utvecklingen, man kan ständigt göra förbättringar/underhåll.
- **Inför mätbara användbarhetsmål.** Låt kravspecifikationen innehålla mätbara användbarhetsaspekter, t. ex. att söka fram ett visst ärende skall inte ta mer än 6 sekunder i anspråk, den procent fel som användaren gör skall understiga 3. Vilka målen är i sig är kanske inte så viktigt som det faktum att användbarheten synliggörs såsom viktig i organisationen. I vissa organisationer har man gått så långt att man vill låta användbarhetsmålen, likväl som andra affärsmål, styra den bonus som de anställda får, allt i syfte att styra verksamheten mot användbarhet.
- **Undvik möten.** Låt inte möten bli ett forum för information och diskussion i stora grupper. Ett effektivt arbetsmöte skall sällan ha mer än 5–7 deltagare. Om man är fler bör man dela in sig i mindre grupper. Information till organisationen kan spridas på annat sätt än genom möten.

10

Referenser och bibliografi

BEYER H., HOLTZBLATT K., (1998), *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufman, San Francisco.

BOEHM B., (1976), *Software Engineering*, IEEE Transactions on Computers, Vol. C25, No. 12, pp. 1226-1241.

BOEHM B., (1988), *The Spiral Model of Software Development and Enhancement*, IEEE Computer, Vol. 21, No 5, pp. 61-72.

BORÄLV E., GÖRANSSON B., *A Teleradiology System Design Case*, in Gerritt van der Veer, Austin Henderson, Susan Coles: Designing Interactive Systems: Processes, Practices, Methods and Techniques. DIS'97 Conference Proceedings of the ACM Special Interest Group in Computer-Human Interaction (SIGCHI) in co-operation with the International Federation for Information Processing (IFIP WG 13.2), Amsterdam August 18-20, pages 27-30.

BUDDE R., KAUTZ K., KUHLENKAMP K., Z¸LLIGHOVEN H., (1992), *Prototyping - An Approach to Evolutionary System Development*, Springer-Verlag, Berlin Heidelberg.

CARROLL J. M., ROSSON M. B. (1985), *Usability specifications as a tool in iterative development.*, in H. R. Hartson (ed.), Advances in human-computer interaction, Norwood, New Jersey.

COOPER, A., (1999), *The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity*, SAMS, Indianapolis, Indiana, ISBN-0-672-31649-8.

GOULD, J.D., & LEWIS, C. (1983), *Designing for usability*, in CHI'83 Conference Boston, Mass, Human factors in computing systems edited by Ann Janda, Amsterdam North-Holland.

GOULD, J.D., & LEWIS, C. (1985) Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM, Vol. 28, No. 3,* pp. 300-311.

CRAMPTON SMITH G., TABOR P. (1996), *The Role of the Artist-Designer*, in Winograd T., Bringing Design to Software, ACM Press, New York, New York.

GULLIKSEN, J., (1996), *Case handling models as a basis for information system design.* In C. A. Ntuen & E. H. Park (Eds.) Human Interaction with complex systems-II. Norwell, MA: Kluwer.

GÖRANSSON B., SANDBÄCK T., (1999), *Usability designers improve the user-centred design process*, i proceedings för INTERACT'99, Edinburgh, UK.

HOUDE S., HILL C., (1997), *What do prototypes Prototype?*, i Helander M. et. al., Handbook of Human-Computer Interaction, Elsevier, Amsterdam.

ISO 9241-11, (1998), *Ergonomic requirments for office work with visual display terminals (VDTs)* – Part 11: Guidance on usability, first edition 1998-03-15, ref. number ISO 9241–11:1998(E), International Organization forn Standardization, Geneve.

ISO 13407, (1999), *Human-centred design processes for interactive systems*, International Organization forn Standardization, Geneve.

KATZEFF C. & SVÄRD P.O. (1995), *Användbarhet i Praktiken: En enkätstudie*. SISU rapport nr. 95:20, Svenska Institutet för Systemutveckling.

KRUCHTEN PHILIPPE, (1998), *The Rational Unified Process—An Introduction*, Addison Wesley Longman Inc., Reading, Mass., USA.

LÖWGREN, J., STOLTERMAN, E., (1998), *Design av informationsteknik – materialet utan egenskaper*, studentlitteratur, Lund, ISBN 91-44-00681-0.

MACLEAN A, YOUNG R.M, BELLOTTI V.M.E, MORAN T.P, (1991), *Questions, options, and criteria: elements of design space analysis*, Human-Computer Interaction 6.

MAYHEW DEBORAH J., (1999), *The usability engineering lifecycle, a practitioner's handbook for user interface design*, Morgan Kaufmann Publishers Inc., San Francisco, CA.

MICROSOFT (1995), *The Windows interface guidelines for software design*, Redmond, WA, Microsoft Press.

MIJKSENAAR P., (1997), *Visual Function*, 010 Publishers, Rotterdam, The Netherlands.

NIELSEN J., (1993a), *Usability Engineering*, Academic Press Inc., San Diego

NIELSEN J., & MACK R.L., (1994), *Usability Inspection Methods*, John Wiley & Sons Inc.

NORMAN D.A. (1986) *Cognitive Engineering*, in D. A. Norman & Draper (eds.) User Centred System Design, Lawrence Erlbaum Associates Inc., Hillsdale, New Jersey.

OLSON E., (1999), *Providing design knowledge to system developers by domain-specific style guides*, Licentiate thesis, Dept. of Human-Computer Intercation, Information Technology, Uppsala University.

POLTROCK S. E., GRUDIN J., (1994), *Organizational Obstacles to Interafce Design and Development: Two Participant—Observer Studies*, ACM Transaction on Computer-Human Interaction, vol. 1 num. 1, sidorna 52-80.

PREECE J., ROGERS Y., SHARP H., BENYON D., HOLLAND S. & CAREY T., (1994), *Human-Computer Interaction*, Addison Wesley Publishing Company, Wokingham, England.

SCHÖN DONALD, (1983), *The Reflective Practioner: How Professionals Think in Action*, New York, Basic Books.

STANDISH GROUP, (1995), *CHAOS report*, tillgänglig via webben http://www.standishgroup.com/chaos.html.

STAPLETON JENNY, (1997), *DSDM – Dynamic Systems Development Method*, Addison Wesley Longman Limited, Essex, England.

TELEDOKS ÅRSBOK 2000, (1999), *Teledok-rapport 130*, Stockholm.

WIGANDER K-O., SVENSSON Å., SCHOUG L., RYDIN A., DAHLGREN C., (1979), *Strukturerad Analys och Konstruktion av informationsbehandlingssystem*, Studentlitteratur Lund.

WINOGRAD T., BENNETT J., DE YOUNG L., HARTFIELD B. (eds), (1996), *Bringing Design into Software*, Addison-Wesley. ■ [slut]