Principer och metoder för systemutveckling

NIKLAS HALLBERG, SOFIE PILEMALM, LARS WESTERDAHL, ERLAND JUNGERT,
HENRIK ERIKSSON, LINA ANDERSSON, FREDRIK LINDMARK

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1000 anställda varav ungefär 800 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
Informationssystem
Box 1165
581 11 Linköping

Tel: 013-37 80 00
Fax: 013-37 81 00

www.foi.se

FOI-R--2628--SE
ISSN 1650-1942

Användarrapport
December 2008

Informationssystem

Niklas Hallberg, Sofie Pilemalm, Lars Westerdahl,
Erland Jungert, Henrik Eriksson, Lina Andersson
Fredrik Lindmark

Principer och metoder för systemutveckling

Titel	Principer och metoder för systemutveckling
Title	Principles and methods for systems development
Rapportnr/Report no	FOI-R--2628—SE
Rapporttyp Report Type	Användarrapport User report
Månad/Month	December/December
Utgivningsår/Year	2008
Antal sidor/Pages	64 p
ISSN	ISSN 1650-1942
Kund/Customer	Försvarsmakten/Swedish Armed Forces
Forskningsområde Programme area	7. Ledning med MSI 7. C4I
Delområde Subcategory	71 Ledning 71 Command, Control, Communications, Computers, Intelligence
Projektnr/Project no	E7140
Godkänd av/Approved by	Martin Rantzer

FOI, Totalförsvarets Forskningsinstitut
Avdelningen för Informationssystem
Box 1165
581 11 Linköping

FOI, Swedish Defence Research Agency
Information Systems
Box 1165
SE-581 11 Linköping

Sammanfattning

Att utveckla ledningssystem är en komplex verksamhet, som är kostnadskrävande och som ofta inte ger ett resultat som motsvarar ställda förväntningar. För att erhålla mer ändamålsenliga ledningssystem till lägre kostnader har Försvarsmakten (FM) och FMV i samverkan med bland annat FOI initierat arbetet med att utveckla nya former för att utveckla och införskaffa ledningssystem. Flera arbeten pekar på att det finns flera områden inom FM ledningssystemutveckling som kan utvecklas vidare. Samtidigt gäller de ”gamla” sanningarna inom systemutveckling, vilka måste beaktas vid införande t av nya utvecklingsansatser. Exempel på dessa är att nyttja användarna, genomföra en grundlig kravhantering och en tydlig utvecklingsprocess. Denna rapport beskriver ett antal begrepp, principer och metoder för systemutveckling vilka måste beaktas när FM och FMV övergår till modellbaserad utveckling.

Nyckelord: Systemutveckling, Begrepp, Utvecklingsprocess
Utvecklingsprinciper, Utvecklingsmetoder

Summary

To develop command and control systems is a complex activity, which is cost consuming, and in many cases do not lead to results that answer to established expectations. In order to arrive at more useful command and control systems at a lower cost, the Swedish Defence (FM) and FMV in collaboration with among others FOI, initiated the work to develop new forms to develop and acquire command and control systems. Several studies demonstrate that there are several areas within FM command and control systems development that can be developed. At the same time the “established truths” in general systems development still apply and must be taken into account when introducing new approaches to development. Examples of those are user participation, to performed an adequate requirements engineering, use of a well-established systems development approach This report describes a number of concepts, principles and methods to systems development, which must considered when FM and FMV goes over to model based development.

Keywords: Systems development, Concepts, Development process, Development principles, Development methods.

Innehållsförteckning

1 Inledning

Utveckling av ledningssystem är en komplex verksamhet, som innehåller många utmaningar som måste hanteras (Hallberg, Pilemalm & Westerdahl 2008). Försvarsmakten (FM) lägger nu ner omfattande resurser på att utveckla verksamheten för att ta fram ledningssystem. Höga förväntningar ställs på att ”modellbaserad utveckling” och FMA (Försvarsmaktens Arkitektur) skall komma till rätta med en del av de svårigheter som tidigare funnits.

Inom systemutvecklingsområdet är svårigheterna med att utveckla komplexa system sedan länge ett välkänt fenomen (Collin, 2003; Chaos, 1995, Brooks, 1995). Andelen så kallade lyckade projekt är låg, liksom nyttjandegraden av funktioner och finesser i tekniska system (Karlsson, 1998). Att ta fram rätt system, till rätt pris och i rätt tid är en stor utmaning (Collin, 2003; Brooks, 1995). Det finns dock ingen brist på utvecklingsansatser, aspekter, koncept och principer för systemutveckling (Sommerville & Sawyer, 1997). Tvärtom, dessa finns i överflöd. Det finns även ett antal vedertagna ”sanningar” som gäller oavsett vilken ansats som väljs.

Denna rapport är den andra från det FoT-finansierade projektet *Arkitekturbaserad ledningssystemsutveckling*. Arbetet som ligger till grund för rapporten syftar till att beskriva ett antal utvecklingsprinciper och metoder samt erfarenheter av systemutveckling som behöver inarbetas, beaktas och förstås vid FM/FMVs etablering av exempelvis modellbaserad systemutveckling.

1.1 Syfte

Syfte med denna rapport är att beskriva ett antal utvecklingsprinciper och metoder samt erfarenheter som är viktiga i systemutveckling, och som bör beaktas och förstås vid utveckling av verksamheter relaterade till ledningssystemutveckling. Detta oavsett vilken ansats för utveckling som anammas.

1.2 Läsanvisningar

Rapporten bör läsas på följande sätt:

Kapitel 2 presenterar bakgrunden till rapporten och motiverar den. Dessutom ges en övergripande projektbeskrivning.

Kapitel 3 beskriver översiktligt genomförandet av studien som ligger till grund för rapporten.

Kapitel 4 beskriver översiktligt den behovsanalys som ligger till grund för stora delar av resultatet, det vill säga de systemutvecklings- principer, begrepp, aktiviteter och metoder som beskrivs i efterföljande kapitel. Behovsanalysen beskrivs i sin helhet i Hallberg, Pilemalm & Westerdahl (2008).

Kapitel 5 till 8 beskriver resultaten - de utvecklingsprinciper, begrepp, aktiviteter och metoder som är centrala i utveckling av system. När man läser detta kapitel och de efterföljande är det viktigt att beakta följande:

- De beskrivna principerna, begreppen, aktiviteterna, och metoderna är det sammanlagda resultatet baserat på en vedertagen systemutvecklingspraxis och den specifika behovsanalys som utförts inom ramen för projektet. Resultatet bygger alltså på forskarnas egna erfarenheter, noggranna litteraturstudier (generell systemutveckling) och en konkret studie (kontexten ledningssystem i FM). I resultatkapitlen beskrivs principer, aktiviteter och metoder c generellt med exempelreferenser till de behov som erhållits genom behovsanalysen och som relaterar till användningen av respektive princip, metod och aktivitet. Begreppen däremot saknar exempelreferenser eftersom de är just begrepp, inte identifierade behov.
- Visa aspekter som beskrivs i rapporten kan ses principer, begrepp, aktiviteter, och/eller metoder. Intressentanalys är exempelvis både en viktig utvecklingsprincip och en utvecklingsaktivitet. Därför skrivs visa aspekter en del saker flera gånger men med olika syfte.
- Rapporten och framför allt resultatkapitlen ska ses som en översiktligt ”uppslagsbok” för ge förståelse kring centrala aspekter och fenomen inom systemutveckling. Rapporten tillhandahåller en rad beskrivningar som bidra till detta.

2 Bakgrund

Följande kapitel beskriver tanken bakom modellbaserad utveckling och projektet *Arkitekturbaserad ledningssystemutveckling*.

2.1 Modellbaserad utveckling

Komplexiteten i informationssystem ökar kontinuerligt och har gjort så under en relativt lång tid. Systemen blir inte bara större utan de kopplas också samman med andra system, som inte alltid kunde förutses under deras utveckling. Överblicken över helheten blir ofta mer fragmenterad viket gör det svårare att ställa rätt krav och att utveckla effektiva system. Den yttersta konsekvensen, utöver system som inte levererar det en kund vill ha, blir att utvecklings- och underhållskostnaderna skenar iväg. Det finns en strävan efter att låta utvecklingen bli behovsdriven snarare än lösningsdriven. Det vill säga att i första ta reda på ”vad som skall göras”, för att därefter bestämma ”vad som behövs för att göra det”. Det har dock alltid varit betydligt lättare att göra tvärtom, ”att bygga det som går och därefter fundera på vad det skall användas till”.

Ett nätverksbaserat och komponentbaserade system befinner sig nästan alltid i någon form av förändring. Programvaror uppdateras, förutsättningarna förändras för tillämpningen i någon del, medan andra delar föråldras och behöver bytas ut. Anledningarna kan vara många men grundproblemen består. Det behövs en utvecklingsmodell med vars hjälp det är möjligt att beakta system som en helhet likväl som att kunna se flera aspekter av systemet på en mer detaljerad nivå. Till detta kommer att det också behövs förmåga till spårbarhet i olika sammanhang.

Modellbaserad utveckling avser att ge förutsättning till spårbarhet från fattade beslut till utvecklade system. Detta inkluderar spårbarhet från strategiska och operativa krav på högsta nivå till taktiska krav på förband, lämpliga förbandsstrukturer och materielsystem för att realisera önskade förmågor. Den ger också stöd för hänsynstagande av hur enskilda system påverkar operativa förmågor samt beroende mellan olika system.

Mjukvaruindustrin har varit en föregångare i att ta till sig den modellbaserade utvecklingsmetodiken. Det är främst nuvarande ansatsers tillkortakommanden i att hantera komplexitet och krav på snabb utveckling samt ökade kostnader som varit drivande.

En modell uttrycker en helhet men det är sällan som hela bilden är intressant. Oftast är man intresserad av en aspekt av modellen – en vy. Det är i vyn, som ett utvecklingsteam kan fokusera på sina problemställningar samtidigt som ett annat

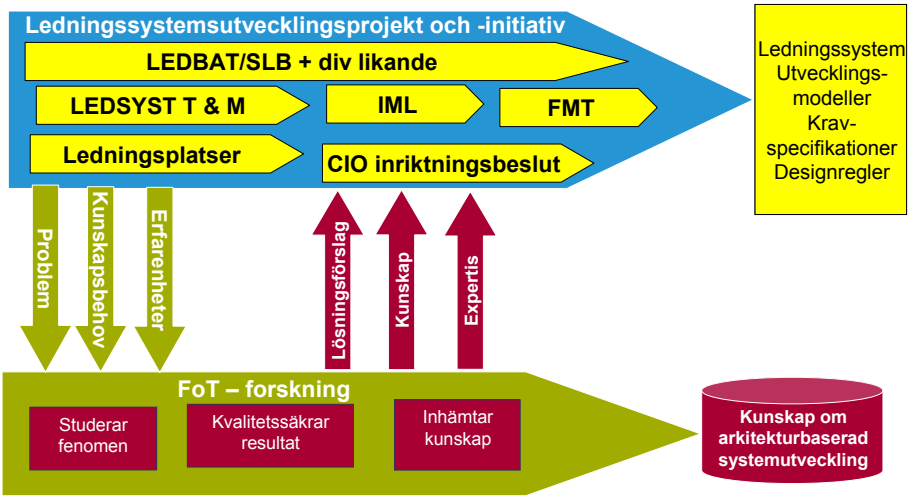
team kan göra motsvarade i en annan vy. Tillsammans arbetar dock teamen i samma modell vilket gör att helheten blir sammanhängande, vilket ger flera positiva effekter. Modellbaserad utveckling under dessa omständigheter förutsätter verktygsstöd och gemensamma datalager.

Modellbaserad utveckling förutses ge:

- Ökade spårbarhet från önskad förmåga, via system realisering, till faktisk förmåga.
- Lägre utvecklings-, anskaffnings- och driftkostnader.
- Lägre personalkostnader
- Kortare utvecklingstider.
- Ökad andel lyckade projekt.
- Stöd för strategisk verksamhetsplanering och materielanskaffning.
- Ökad flexibilitet med avseende på möjlig verksamhet och organisationsstruktur.
- Möjlighet att nyttja kvalitetsmått på verksamhetsprocesser och information.

2.1.1 Arkitekturbaserad ledningssystemsutveckling

Projektet *Arkitekturbaserad ledningssystemsutveckling* syftar till att tillhandhålla FM stöd avseende arkitekturbaserad utveckling av ledningssystem. Målet med projektet är att utveckla kunskap och kompetens avseende arkitekturbaserad ledningssystemsutveckling, som är vetenskapligt kvalitetssäkrad och praktiskt förankrad. Detta skall ske genom att projektets medarbetare aktivt medverkar inom olika ledningssystemutvecklingsprojekt och initiativ inom FM. Detta för att inhämta kunskap avseende svårigheter, kunskapsbehov och erfarenheter och sedan använda denna kunskap som grund för att genomföra forskningsverksamhet inom FoT-projektet. Därefter kan denna kunskap återföras med avseende på möjliga lösningar till andra projekt och liknade verksamheter (Figur 1). Under projektet skapas en databas med kvalitetssäkrad information avseende arkitekturbaserad ledningssystemsutveckling. Denna rapport beskriver resultatet från projektets initiala fas, som omfattar behovsanalys och kunskapsinventering.



Figur 1. Beskrivning av relationen och informationsflödet mellan FMs lednings-systemutvecklingsprojekt, andra initiativ och FoT-projektet.

3 Genomförande

Genomförandet av studien baserades på två aktiviteter; (1) behovsanalys och (2) identifiering av principer och metoder viktiga för att tillfredställa dessa behov.

3.1 Behovsanalys

Initialt genomfördes intervjuer med 16 respondenter från FM, FMV och FOI. Dessa representerade skilda nivåer, områden och organisationer som är involverade i FM ledningssystemutveckling. Syftet vara att identifiera vilket behov av stöd och förbättringar som föreligger i FM ledningssystemutveckling. Den information som erhölls under intervjuerna analyserades med avseende på att identifiera behoven. Dessa behov kategoriserades och strukturerandes. Genomförandet och det fullständiga resultatet av behovsanalysen beskrivs i Hallberg, Pilemalm och Westerdahl (2008), en sammanfattning av behovsanalysens resultat ges i Kapitel 4.

3.2 Identifiering av principer och metoder

Med utgångspunkt från behoven och erfarenheter avseende systemutveckling identifierades ett antal utvecklingsprinciper som är viktiga vid utveckling av system samt begrepp centrala för systemutveckling, en generisk utvecklingsprocess samt ett antal utvecklingsmetoder. Detta har skett genom litteratursökningar, dokumentstudier och en generell, erfarenhetsbaserad kunskapsinventering av de deltagande forskarnas olika kompetensområden. Dessa dokumenteras i denna rapport.

4 Behovsanalys

Under projektets initiala skede genomfördes en behovsanalys avseende vilket stöd, vilka förutsättningar och aktiviteter inom FMs verksamhet som behövs för att möjliggöra utveckling av ledningssystem (Hallberg, Pilemalm & Westerdahl, 2008). I detta ingår att bestämma vad som är viktigt för att lyckas med utvecklingen och för att slutresultatet, det vill säga ledningssystemet, skall bli så ändamålsenligt som möjligt. De identifierade behoven avses ge en grund för det fortsatta arbetet i projektet, men även att vara till nytta för andra initiativ som syftar till att effektivisera och förbättra FMs utveckling av ledningssystem. Behov anges nedan för att kunna refereras i senare kapitel (Tabell 1).

Tabell 1 Identifierade behov avseende stöd, förutsättningar och aktiviteter inom FMs verksamhet för att möjliggöra utveckling av ledningssystem. Behoven samt genomförandet av behovsanalysen finns fullständigt beskrivet i Hallberg, Pilemalm och Westerdahl (2008).

1 Projektledning och -styrning
Behov 1:1: Tydlig styrning och koordinering av ledningssystemsutvecklingen
Behov 1:2: Tydlig roll och ansvarsfördelning
Behov 1:3: Effektiv beslutsprocess
Behov 1:4: Tydliga målsättningar i ledningssystemsutvecklingen
Behov 1:5: Fokus på givna regler
Behov 1:6: Synkronisera aktiviteterna i ledningssystemsutveckling
Behov 1:7: Koordinera utvecklingsinitiativ över projektgränser
Behov 1:8: Samverka över produktions¬områden
Behov 1:9: Långsiktighet i ledningssystemsutveckling
Behov 1:10: FMV att ta del av FMs beslut
Behov 1:11: Prioritera förmågor
2 Ledningsutvecklingsprocess
Behov 2:1: Gemensam ledningssystemsutveckling
Behov 2:2: Sammanhållen och holistisk ledningssystemsutveckling
Behov 2:3: Tydlig och väl definierad process för ledningssystemsutveckling
Behov 2:4: Kostnadseffektiv ledningssystemutveckling

Behov 2:5: Bättre integration av ledningsmetodik- och teknikutveckling
Behov 2:6: Bygga ledningssystem evolutionärt
2.1 Behovsanalyser
Behov 2.1:1: Identifiera intressenter
Behov 2.1:2: Identifiera de verkliga behoven
2.2 Kravhantering
Behov 2.2:1: Korrekt kravbild
Behov 2.2:2: Specificera krav
Behov 2.2:3: Specificera krav på COTS
Behov 2.2:4: Realistiska krav
Behov 2.2:5: Modifiera krav efterhand
Behov 2.2:6: Konsistenta kravspecifikationer
Behov 2.2:7: Prioritera krav
2.3 Design
Behov 2.3:1: Säkerställa samma/liknande behov får samman tekniska lösning
2.4 Modellering
Behov 2.4:1: Modeller i utvecklingsprocessen
Behov 2.4:2: Modeller för användarmedverkan
Behov 2.4:3: Beskrivningar av användningskontext
Behov 2.4:4: Beskriva förmågor
2.5 Systemanskaffning
Behov 2.5:1: Jobba i längre cykler
Behov 2.5:2: Förenklad auktoriseringsprocess
Behov 2.5:3: Systemanskaffning som är anpassad till de snabba förändringarna i FM
Behov 2.5:4: Stöd för upphandling mot industrin
2.6 Utvärdering
Behov 2.6:1: Utvärdera levererade produkter och resultat
Behov 2.6:2: Utvärdera ledningssystem som används

Behov 2.6:3: Säkra kvalitén hos koncept i skarpa övningar
3 Utvecklingsmetoder
Behov 3:1: Gemensamma utvecklingsmetoder
Behov 3:2: Enkla utvecklingsmetoder
Behov 3:3: Utvecklingsmetoder för att nyttja användarmedverkan
4 Verktygsstöd
Behov 4:1: Adekvat verktygsstöd
Behov 4:2: Kravställa verktygsstöd
5 Kompetens
Behov 5:1: Kompetens avseende ledningssystemsutveckling
Behov 5:2: Tillgång till teknisk kompetens
Behov 5:3: Tillgång till militär kompetens
Behov 5:4: Ha programmerare när knutna till verksamheten
Behov 5:5: Systemutvecklingskompetens som leder arbetet
Behov 5:6: Kompetens för verksamhetsmodellering
Behov 5:7: Kompetens för kravhantering
Behov 5:8: Utbildning avseende utveckling av ledningssystem
Behov 5:9: Kompetensöverföring mellan människor
5.1 Användarmedverkan
Behov 5.1:1: Kompetens kring användarmedverkan
Behov 5.1:2: Genomföra realistiska användarstudier
Behov 5.1:3: Medverkan av representativa användare
Behov 5.1:4: Kontakt med många olika användare
Behov 5.1:5: Involvera användare på ett tidigt stadium
Behov 5.1:6: Aktiv användarmedverkan
Behov 5.1:7: Kontinuerlig kontakt med användare
Behov 5.1:8: Återkoppling från användare
Behov 5.1:9: Hålla användarrepresentanterna motiverade
Behov 5.1:10: Tydliggöra skillnader mellan prototyper och verkliga produkter

Behov 5.1:11: Frigöra FM personal som användarrepresentanter
5.2 Utnyttja forskning
Behov 5.2:1: Kunna nyttja forskning
6 Spårbarhet
Behov 6:1: Spårbarhet mellan verksamhetens behov, kraven, kravställare, realiseringar
Behov 6:2: Spårbarhet mellan efterfrågad förmåga till lösningar och faktiskförmåga
Behov 6:3: Spårbarhet mellan teknisk utveckling och tillämpningen
Behov 6:4: Tydliggöra hur tekniken stödjer ledningen
7 Samverkan
Behov 7:1: Samverkan mellan FM och FMV
Behov 7:2: Samverkan mellan ledningsområdet och de andra enheterna i FM
Behov 7:3: Väl fungerande samverkan mellan FM, FMV och industrin
Behov 7:4: Samverkan FM och FMV vid kravspecificering
Behov 7:5: Samarbete mellan arme, flyggvapen, marin inom FM såväl som FMV
Behov 7:6: FMV att jobba mer med den omliggande världen
Behov 7:7: Kontinuerlig samverkan i utvecklingsgrupper
Behov 7:8: Bilaterala samarbetsavtal inom och utom EU
8 Helhetsbild
Behov 8:1: Beskriva systems externa egenskaper och dess sammanhang
9 Säkerhet
Behov 9:1: Integrera säkerheten redan tidigt i utvecklingsarbetet
Behov 9:2: Specificera krav på säkerhet balanserat mot verksamhetens behov
10 Snabbhet i utvecklingen
Behov 10:1: Driva viss utveckling teknikdriven
Behov 10:2: Effektivare utveckling av ledningssystem
Behov 10:3: Kortare utvecklingscyklar för ledningssystem

11 Utnyttja erfarenheter
Behov 11:1: Nyttja gjorda erfarenheter
Behov 11:2: Nyttja existerade modeller
Behov 11:3: Återanvända grundläggande krav
Behov 11:4: Återanvända utvecklade funktioner
Behov 11:5: Återanvända utvecklade applikationer
12 Utvecklingsteam
Behov 12:1: Sammansvetsade utvecklingsteam
Behov 12:2: Långsiktighet avseende personal i utvecklingsprojekt
Behov 12:3: Formella, integrerade grupper för ledningssystemutveckling
13 Dokumentation
Behov 13:1: Gemensamt dokumentationsformat
Behov 13:2: Bättre dokumentationsformat för krav på informationssystem
Behov 13:3: Bättre dokumentationsformat för att beskriva förmågor hos förband
14 Systemkarta
Behov 14:1: Kartlägga vilka ledningssystem som behöver kunna kommunicera med varandra
Behov 14:2: Interoperabilitet mellan ledningssystem
15 Resurser
Behov 15:1: Mer resurser för ledningssystemutveckling
Behov 15:2: Vidareutveckla faciliteter för experiment, simulering och träning
16 Medvetenhet
Behov 16:1: Förståelse för att det är användningen av teknik som ger effekt
Behov 16:2: Insikten att dåliga IT system förstör fungerade verksamhet
17 Övrigt
Behov 17:1: Nyttja kreativitet i utvecklingsarbetet
Behov 17:2: Kostnadseffektiv drift av ledningssystem
Behov 17:3: Nyttiggöra system som utvecklas vid exempelvis FOI
Behov 17:4: Se över FMs utvecklingsorganisation

Behov 17:5: Prioritera metod över teknik
Behov 17:6: Gå runt restriktioner runt lagen om offentlig upphandling
Behov 17:7: Strategiska planer för att köpa upp industrin
Behov 17:8: Dela risker med leverantörer
Behov 17:9: Informationsspridning runt service-orienterade ledningssystem
Behov 17:10: Informationsspridning om ledningssystem på tre nivåer
Behov 17:11: Utveckla sig mot civil marknad
Behov 17:12: Bättre anpassade kommunikationsdelar inom ledningssystem

5 Utvecklingsprinciper

Detta avsitt beskriver ett antal centrala *principer* som är viktiga för att lyckas med systemutveckling. Referenser anges till behoven i Tabell 1.

5.1 Väldefinierad utvecklingsprocess

Det är viktigt att arbeta enligt någon väldefinierad utvecklingsprocess, med tydliga mål och delmål vid all systemutveckling (Kasser, 2007). Processen ska vara väl definierad, förstådd och accepterad av berörda parter. Exakt vilken process som väljs är inte det viktigaste. Huvudsaken är att den är väl beprövad, att alla vet och accepterar att följa den. I annat fall är risken stor att tid och resurser satsas på sådant som inte har någon positiv effekt på slutresultatet. Det vill säga det måste finnas ett syfte med varje aktivitet, att den skall bidra till att uppnå slutmålet.

Exempel på relaterade behov:[2:3],[2:1], [3:1]

5.2 Kompetens

Tillgång till personal med rätt kompetens måste finnas under hela utvecklingen. Det är viktigt att kontinuiteten i detta avseende bibehålls. Ständig omsättning av personer är en stor risk. Vilket medför att det är viktigt att de som är involverade i utvecklingen ”belönas” för detta samt att de motiveras att kontinuerligt vidareutveckla sig.

Exempel på relaterade behov:[5:1],[12:1], [12:2]

5.3 Tydligt ansvar

Den grupp som får ansvaret för att genomföra ett utvecklingsprojekt måste få tydliga mandat för detta. Väl genomarbetade kontrakts- och kommunikationsformer mellan beställare och utförare är en förutsättning för att lyckas genomföra systemutveckling med gott resultat.

Exempel på relaterade behov:[1:1],[1:2], [12:2]

5.4 Tillgång till användarrepresentanter

Tillgång till användarrepresentanter är en viktig aspekt av att ha tillgång till adekvat kompetens. Det är användarna som är specialister på

användningssituationen. De kan bidra både med att beskriva den, kommentera beskrivningar av den och de är en viktig källa till information som kan nyttjas för att identifiera behoven av systemet. Användarrepresentanter är också viktiga då det gäller att validera olika lösningar. Däremot skall de inte användas till rent systemutvecklingsarbete, exempelvis att ta fram UML-modeller, mata in krav etc. Det är inte heller de som är specialister på att specificera krav.

Exempel på relaterade behov:[5:1:1-5:1:11],[3:2], [3:3]

5.5 Tydliga systemgränser

Innan några andra utvecklingsaktiviteter initieras är det viktigt att tydligt definiera vilket system som skall utvecklas och vad som tillhör systemomgivningen (Sommerville, 2001). Tydliga gränser som förstås av samtliga involverade i utvecklingen är viktigt.

Exempel på relaterade behov:[8:1]

5.6 Kontextuell förståelse

Förståelse av den omgivning (kontext) som systemet skall användas i är viktigt. Det är genom aktörerna i kontexten som kraven på systemet måste identifieras. Detta innefattar såväl tänkta användare som antagonistiska aktörer där de senare inte är önskvärda eftersom de kan använda och/eller manipulera systemet på något icke önskvärt sätt. När det gäller ledningssystem är omgivningen så komplex att det inte räcker med någon enkel beskrivning eller modell för att beskriva denna. Det är i dessa fall som ”Enterprise Architecture” ses som en framkomlig väg (McGovern, Ambler, Stevens, Linn, Jo, & Sharan, 2003).

Exempel på relaterade behov:[5:3], [5:1:3], [5:1:5]

5.7 Tydlighet avseende begrepp

Inom systemutvecklingsområdet används många begrepp, dessvärre inte entydigt. Detta kan leda till missförstånd som får allvarliga konsekvenser för utvecklingen. Det är viktigt att åtminstone inom samma projekt och verksamhet ha en delad syn på vad begreppen står för.

Exempel på relaterade behov:[2:1], [6:1], [13:1]

5.8 Intressentanalys

Att förstå vilka intressenterna till ett system är och vilka deras så kallade ”painpoints” är absolut nödvändigt för att kunna motivera utvecklingen. Det vill säga, vilka problem och brister intressenterna upplever i sin verksamhet. Detta utgör även grunden för kommunikationsplanen.

Exempel på relaterade behov:[2:1:1], [2:1:2]

5.9 Behovsanalys

En gedigen och grundläggande behovsanalys är nödvändig för att inte outtalade behov skall gå förlorade (Kano, 1995). Vilka behov som skall ligga tillgrund för utvecklingen bör prioriteras hårt (Karlsson, 1998). Det är viktigt att tillräcklig tid avsätts för behovsanalys då misstag och brister blir kostsamma att rätta till senare under utvecklingen (Collin, 2003).

Exempel på relaterade behov:[2:1:1], [2:1:2], [1:11, [2:2:1], [6:1]

5.10 Kravhantering

Bristande kravhantering anses vara grunden till de flesta och mest kostsamma bristerna i system (Young, 2001). Kravhantering kan ses som den ”brygga” som binder samman behoven med design lösningen. Svag hantering av kraven innebär att systemet inte kommer att möta avsedda behov och den önskade effekten av systemet uteblir. Det är viktigt att tillräckliga resurser avsetts för hanteringen av krav då misstag och brister blir kostsamma att rätta till senare i projektet (Collin, 2003).

Exempel på relaterade behov:[2:2:1-2:2:7]

5.11 Designa för användning

Det är ofta svårt att avgöra vad som är en ”bra” design utifrån ett användningsperspektiv (Gulliksen & Göransson, 2002). Det är betydligt lättare att peka på dåliga lösningar, det vill säga dålig design. Att utvärdera prototyper av systemets gränssnitt med stöd av användarrepresentanter bidrar till att snabbt få underlag för att korrigera misstag i designen. Det finns vidare ett antal generella och specifika rekommendationer som bör beaktas för designen beroende på för vad och för vem systemet designas.

Exempel på relaterade behov:[16:1], [16:2]

5.12 Spårbarhet

För att bedöma om det är ”rätt” system som utvecklas krävs spårbarhet; exempelvis för att motivera ”krav mot behov” och ”design mot krav” (Hallberg, 1999). Spårbarhet måste upprätthållas genom hela utvecklingskedjan och bör vara dubbelriktad. Det vill säga, det skall vara möjligt av avgöra exempelvis vilka behov som ett krav härrör från samt vilka olika system funktioner och egenskaper som relaterar till ett givet krav. Spårbarhet måste också finnas mellan olika behov, mellan olika krav etc (Hull, Jackson & Dick, 2005). Det vill säga hur exempelvis olika krav relaterar till varandra.

Exempel på relaterade behov:[2:2], [2:2:1]

5.13 Prioritering

Att utveckla system som klarar allt eller ens många saker är omöjligt. Det viktigt att resurserna fokuseras på sådant som är viktigast. Prioriteringar måste kontinuerligt ske, från vilka intressenter som skall få påverka utformningen av system till val av lösningar. Behov och krav är centrala prioriteringar. Avsaknad av prioriteringar medför att utvecklingen riskerar att förlora fokus på det som är viktigt.

Exempel på relaterade behov:[2:2:7], [10:2], [10:3]

5.14 Kontinuerlig kvalitetssäkring

Resultatet av systemutvecklingen blir sällan eller aldrig bättre än kvalitén på indata och på hanteringen av denna. Det är inte möjligt att upprätthålla perfekt spårbarhet och felaktiga beslut tas oundvikligen under utveckling. För att minska effekterna av detta krävs en kontinuerlig kvalitetssäkring av utvecklingsprocessen och del resultaten från de olika aktiviteterna i systemutvecklingen. Det finns många olika tillvägagångssätt för säkra kvaliteten i slutprodukten (Collin, 2003; Sommerville, 2001).

Exempel på relaterade behov:[2:2:1], [17:2]

5.15 Modellering

Att använda sig av olika modellansatser är viktigt i systemutveckling (Hay, 2003; Sommerville, 2001; Friedenthal, Moore & Steiner, 2008) . Sådana modellansatser kan användas för att beskriva och begripliggöra komplexa samband och tänkta lösningar (Hallberg & Fransson, 2001). Modeller kan också användas för att beskriva existerande systemomgivningar, en tänkt framtida omgivning, och system i olika utvecklingsfaser. Det som är av största vikt vid modellering är att som första steg bestämma syftet med modellen och därefter fokusera på framtagandet av denna detta syfte i fokus. Stora modeller blir fort ohanterliga och oanvändbara, Det är också svårt att avgöra vilket som är den viktiga informationen i dessa modeller. Modeller tar ofta lång tid att framställa, och än mer resurser krävs för att underhålla och uppdatera dem. Informationsmodeller har ofta lägre validitet än processmodeller (Hay, 2003).

Exempel på relaterade behov:[2:4:1-2:4:4]

5.16 IT-säkerhet

IT-säkerhetsaspekter måste beaktas tidigt i utvecklingen och IT-säkerhetssystem måste kontinuerligt integreras med övriga systemmoduler (Hallberg & Hallberg, 2006). Det vill säga när de funktionella behoven beaktas måste av även behov av IT säkerhet beaktas, när de funktionella kraven identifieras måste även IT-säkerhetskraven identifieras.

Exempel på relaterade behov:[9:1], [9:2]

6 Begrepp inom systemutveckling

Detta kapitel beskriver och förklarar ett antal *begrepp* som är centrala inom systemutveckling. Det är uppdelat i fyra kategorier av begrepp: Generella begrepp, Beskrivningsbegrepp, Utvecklingsprocessbegrepp och System-uppträdandebegrepp.

Exempel på relaterade behov:[2:1]

6.1 Generella begrepp

De generella begreppen innefattar: System, Ledningssystem, Systemomgivning, Komponent, Vy, Arkitektur, Arkitekturramverk, Ramverk, Systemutveckling.

System är en samling komponenter som organiserats för att åstadkomma en eller flera specifika funktioner (ANSI/IEEE, 2007). System kan vara tekniska, organisatoriska eller kombinationer av dessa. System består av komponenter (som kan vara andra system) och existerar i en systemomgivning. Både komponenter och systemomgivning kan betraktas som system. Vad som är vad avgörs av vilket system som är i fokus.

Ledningssystem är en speciell typ av system som utgör ledningen av ett annat system. Generellt har ledningssystem tre uppgifter (1) samla in information, (2) fatta beslut, och (3) delge information och order. Inom Försvarsmakten, definieras ledningssystem som bestående av människor och tekniska stöd, vilka är organiserade och följer givna arbetsprocedurer. Inom Försvarsmakten finns i huvudsak två typer av ledningssystem, insatsledningssystem och verksamhetsledningssystem. I andra sammanhang ses ledningssystem som det tekniska system som stöder beslutsfattare att genomföra ledning.

System komponent är en del av det betraktade systemet som ses en enhet.

Systemomgivning är den omgivning, kontext, i vilket systemet fungerar eller skall fungerar i.

Systemutveckling är processen att utveckla system. Det existerar en stor mängd olika processer. Några av de mest kända är Vattenfallsprocess, Spiralmodellen, Joint Appalication Development (JAD) (Wood & Silver, 1995), Rapid Application Development (RAD) (Martin, 1991), Business Process Reegineering (BPR) (Davenport, 1993), Soft Systems Methodology (Checkland & Scholes, 2001) och Rational Unifed Process (RUP) (Kruchten, 2004). RUP är den ansats som på senare år fått ett relativt stort intresse och presenters ofta som en systemutvecklingsprocess, begränsad till utveckling av mjukvara. RUP skall

dock snarare ses som ett processramverk. RUP kan nyttjas för utveckling av mer komplexa system och mjukvarusystem (Pilemalm, Lindell, Hallberg, & Eriksson, 2007). Det finns ett antal begrepp som beskriver utvecklingsprocessens genomförande. *Iterativ utveckling* som innebära återkoppling från senare faser till tidigare för att göra omarbetningar exempelvis ta fram en ny version av kravspecifikationen *Inkrementell utveckling* innebär att operativa delar av systemet levereras efter hand och att varje ny version av systemet har en ökad funktionalitet (Larman, 2003). *Evolutionär utveckling* är både iterativ and inkrementell.

Arkitekturbaserad systemutveckling avser utveckling som nyttjar sammanhängande beskrivningar av omgivningen och/eller systemet; det vill säga beskrivningar av deras arkitektur. Detta utifrån ett helhetsperspektiv, avseende såväl dynamiska som statiska egenskaper. Arkitekturen för systemomgivningen nyttjas för att identifiera de behov som det utvecklande system skall tillfredställa samt övriga krav som skall ställas på systemet. Exempel på detta är ”Enterprise Architecture” och dess användning för att styra organisationers IT-stöd. Arkitekturen för systemet som skall utvecklas skall ge en översikt över systemdesignen, utan detaljerade beskrivningar av enskilda komponenter och funktioner. Den skall beskriva hur olika komponenter skall integreras. Arkitekturerna skall konstrueras ackumulativt och iterativt, med utgångspunkt i de behov som finns, hos dess användare och övriga intressenter.

Arkitektur beskriver ett systems väsentliga komponenter, deras relationer till varandra samt till systemets miljö och de intressenter som ska använda systemet (ANSI/IEEE, 2007). En arkitektur bör även beskrivas efter vilka principer systemet har konstruerats och hur dess livscykel hanterats. Alla system har en arkitektur, oavsett om den är formaliserad och medvetandegjord eller ej. Ett systems arkitektur och beskrivningen av denna är dock två skilda företeelser. Beskrivningen av en arkitektur är ofta organiserad efter en samling vyer som var och en beskriver systemet i ett specifikt syfte. En “Enterprise Architecture” är en holistisk beskrivning som innefatta all möjlig information av betydelse för verksamheten, såsom organisation, mål, verksamhetsuppgifter, aktiviteter, relationer, applikationer och infrastrukturer. Att nyttja arkitekturer gör det möjligt att få översikt och styrbarhet av stora komplexa system så att dessa uppträder som flexibla och till situationen anpassningsbara enheter. Att nyttja en arkitekturbaserad ansats är fördelaktig i så väl nyutveckling som vid modifiering och underhåll av system. Det möjliggör att återanvända befintliga system och komponenter för att åstadkomma ny och förbättrad funktionalitet.

Arkitekturramverk används för att utveckla arkitekturer, för olika ändamål och områden. För att tillhandahålla sammanhängande och konsekventa modeller

krävs det ett ramverk som definierar en specifikation för hur modeller kan organiseras och presenteras. För att uppnå detta används någon form av metamodell som bidrar till att med ett enhetligt språk beskriva arkitekturer, ingående element och deras relationer (MODAF, 2007). Det existerar ett antal arkitekturramverk för att utveckla arkitekturer för organisationer (Enterprise Architecture). Eftersom dessa ramverk ofta är generella måste viss anpassning av dessa och deras används göras. Många ramverk har flera gemensamma egenskaper. Det mest kända arkitekturramverk är Zachmans (1987). Andra pågående utvecklingar av arkitekturramverk är Ministry of Defence Architectural Framework (MODAF), NATO Architectural Framework (NAF), Department of Defense Architectural Framework (DoDAF) och The Open Group Architecture Framework (TOGAF) (Schekkerman, 2003; MODAF, 2007). Försvarsmakten har valt att använda MODAF som grund för sitt FM AR (Försvarsmaktens arkitekturramverk).

Vy är en representation av ett system utifrån ett perspektiv av antal aspekter (IEEE, 2000). Vilka vyer som skall användas för att beskriva ett system beror på vilka behov dess intressenter har.

Ramverk är en struktur, som kan hålla samman verktyg, metoder, modeller och processer.

Process är ett antal samman kopplande aktiviteter med syfte att producera något specificerat (Davenport, 1993).

Fas är en del av en process. Det vill säga en process kan vara uppdelad i olika tidsberoende faser.

Aktivitet är ett antal handlingar som konsumerar tid och resurser vars genomförande är nödvändiga för att uppnå ett eller flera mål (IEEE, 2000).

Modell är en abstrakt beskrivning av ett fenomen. Detta fenomen kan vara verkligt eller virtuellt. En *verksamhetsmodell* är en abstraktion som stödjer förståelse av verksamheten, och som därmed utgör ett verktyg för att analysera och kommunicera kring en viss verksamhet. En *prototyp* är en modell avsedd att illustrera uppbyggnaden och funktionaliteten av något, exempelvis ett informationssystem eller en bil.

Intressent är en individ eller en grupp som påverkas eller påverkar det aktuella systemet eller dess utveckling. Intressenter omfattar användare, kunder och systemadministratörer.

Aktör är någon eller något, utanför ett system, som interagerar med systemet. Aktörers interagerade kan vara avsiktligt eller oavsiktligt, med eller utan mål.

Användare är någon eller något som med behörighet avsiktligt interagerar med systemet för att uppnå ett syfte (Sutcliff, 2002). Primära eller direkta användare interagerar direkt med systemet, medan sekundära användare interagerar med systemet via direkta användare (Hallberg, 1999).

Roll svarar mot vissa uppgifter eller –inriktning. Användare och aktörer kan anta ett antal roller. Roller kan användas för att reglera behörighet att utföra funktioner i ett system.

Kunder är individer och organisationer som direkt upphandlar eller medverkar till beslut att upphandla systemet (Sutcliff, 2002).

6.2 Beskrivningsbegrepp

Beskrivningsbegreppen innefattar begrepp som bidrar till att beskriva systemet i dess olika utvecklingsfaser. Dessa innefattar Utsaga, Behov, Krav, Design och Realisering.

Exempel på relaterade behov:[13:1]

Utsaga är ett yttrande som innehåller information av betydelse för det system som skall utvecklas Utsagor uttrycks av intressenter i termer som observationer, antaganden, intresseyttringar etc. De kan innehålla beskrivningar av problem, tankar om möjliga lösningar, krav etc. De kan vara beskrivningar av nuvarande eller önskade situationer, erfarenheter av problem, verksamhetsmål eller visioner De behöver därmed inte vara direkt uttryckta med syfte på det system som skall utvecklas.

Behov representerar något som saknas, är önskvärt och användbart, något som stödjer någon i att genomföra uppgifter och uppnå mål. Behov som är tillfredställa med existerade lösning måste också beaktas när dessa skall ersättas av någon annan lösning. I annat fall kan tillfredställda behov plötsligt bli otillfredsställda. *Användarbehov* är behov som personer har för att uppnå sina mål och lösa sina uppgifter. *Verksamhetsbehov* är behov som organisationer har för att uppnå sina målsättningar.

Krav specificerar vad system skall åstadkomma (Siddiqi and Shekaran, 1996). Krav kan delas in i funktionella och icke-funktionella krav där de funktionella kraven beskriver vad som skall åstadkommas medan de icke-funktionella kraven beskiver vilka egenskaper systemet måste ha.

Design beskiver hur funktioner och egenskaper skall realiseras i systemet. Den beskriver både synliga och osynliga aspekter. Designen är en modell av systemet som utvecklas.

Realiseringen utgörs av faktiska lösningar, för att åstadkomma det designen beskriver.

6.3 Utvecklingsprocessbegrepp

Utvecklingsprocessbegreppen innefattar aktiviteter som genomförs under systemutveckling. Dessa aktiviteter är Kontextanalys, Behovsanalys, Kravhantering, Design, Realisering, Införande, Verifiering, Validering och Exploatering.

Exempel på relaterade behov:[2:2]; [3:1]

Kontextanalys syftar till att analysera den omgivning i vilket ett system skall fungera. Om ett befintligt system skall ersättas eller modifieras är det viktigt att även beakta hur det nuvarande systemet fungerar. *Verksamhetsanalys* är en vanlig form av kontextanalys, med syfte att erhålla förståelse för kontexten i vilken systemet skall användas. Verksamhetsanalys är även grund för att genomföra en verksamhetsförändring. Ett vanligt resultat av kontextanalyser och verksamhetsanalyser är någon form av verksamhetsmodell eller -beskrivning. En ytterligare del i denna typ av analyser är *intressentanalyser*. Deras syfte är att identifiera och beskriva intressenter till systemet som skall utvecklas. Modeller och beskrivningar som resultat av dessa analyser är viktiga som indata för behovsanalyser.

Behovsanalys syftar till att identifiera vilka behov som finns avseende det system som skall utvecklas. Behovsanalyser kräver god insikt i den miljö som systemet skall användas i. Det är viktigt att redan i detta skeende av utvecklingen börja prioritera vilka behov som är viktiga för att systemet skall tillfredställa de slutliga kraven. Resultatet av en väl genomförd behovsanalys skall innefatta de behov som skall ligga tillgrund för den vidare utvecklingen.

Kravhantering syftar till att specificera vad systemet skall åstadkomma, utan att beskriva hur (Siddiqi & Shekaran, 1996). Men kravhantering innefaffatr att besluta vad system inte skall klara, genom att välja bort krav. Aktiviteten innefattar identifiering, analys, dokumentation, och validering av kraven (Wiegers, 2003). I analysen bör en prioritering göras av hur viktiga kraven är att tillfredställa. Resultatet av kravhanteringen utgörs av en kravspecifikation.

Design syftar till att specificera hur system skall realiseras. Detta inkluderar hur gränssnittet skall se ut, vilka tekniker, kommunikationsprotokoll etc som skall användas. Resultatet är en beskrivning av designen.

Realisering syftar till att omsätta designen, som är en modell, till ett fungerande system. Detta inkluderar kodning av programvara och rekrytering av nya kompetenser.

Införande syftar till att införa det utvecklade system i den kontext där det skall användas. När införandet görs ställer detta ofta krav på omgivningen.

Verifiering syftar till att bekräfta att designen alternativt det realiserade systemet motsvarar de krav som specificerats.

Validering syftar till att bekräfta att kraven, designen alternativt det realiserade systemet ger den effekt i omgivningen som avsågs.

Exploatering syftar till att studera om ett existerande system kan nyttjas även för andra ändamål; ge nytta som det ursprungligen ej var avsett för. Det vill säga möta ytterligare behov som det ej från början var avsett att tillfredställa.

6.4 Systemuppträdandebegrepp

Systemuppträdandebegrepp innefattar begrepp som beskriver hur system ser ut, fungerar och uppträder. Dessa begrepp innefattar Funktion, Kapacitet, Förmåga, Egenskap, Tjänst och Användbarhet.

Funktion beskriver den handling som systemet används för.

Egenskaper beskriver det utseende, struktur, form, framträdande och karakteristik som ett system har.

Förmågor beskriver de funktioner som system kan tillhandahålla under givna förutsättningar.

Kapacitet beskriver vilka kvantiteter av ett givet funktionsresultat som ett system kan producera.

Effekt beskriver resultatet av ett systems funktion på omgivning.

Tjänst är en abstrakt beskrivning av hur ett system (*tjänsteproducent*) kan åstadkomma nytta för andra system (*tjänstekonsument*) utan att beskriva hur detta görs. En tjänst är ett standardiserat gränssnitt mellan olika system som syftar till att skapa interoperabilitet, för att möjliggöra modulära och flexibilitet sammansatta konfigurationer av system. Förmåga, kapacitet och effekt är exempel på egenskaper hos tjänster.

Användbarhet är den upplevda och faktiska nytta av systemet som användare erfar. Det vill säga att systemet fyller ett ändamål och samtidigt är tillräckligt enkelt att använda. Om system inte upplevs tillföra någon nytta kommer det att förbli oanvänt. Ett sätt att öka användbarhet är att iterera utvecklingen och involvera användarna.

7 Systemutvecklingsaktiviteter

Det finns ett antal olika sätt att beskriva systemutveckling baserat på ingående *aktiviteter*. I denna rapport beskrivs systemutveckling med hjälp av aktiviteterna Kontextanalys, Behovsanalys, Kravhantering, Design, Realisering, Verifiering och Validering.

7.1 Kontextanalys

Kontextanalys syftar till att analysera den omgivning i vilket det system som skall utvecklas skall fungera i. Detta för att få underlag för att identifiera vilka behov, krav och restriktioner som systemet skall uppfylla. Om det är ett befintligt system som skall vidareutvecklas är det viktigt att även detta ingår i analysen. *Verksamhetsanalys* och *intressentanalyser* är vanliga former av kontextanalys.

7.1.1 Verksamhetsanalys

En väl genomförd verksamhetsanalys ger en grund för att åstadkomma en verksamhetsförändring, oavsett om den baseras på förändrade arbetssätt, organisation eller införandet av ett IT-stöd. Ett vanligt resultat av verksamhetsanalyser är någon form av verksamhetsmodell eller -beskrivning.

7.1.2 Intressentanalys

Intressentanalyser syftar till att identifiera och beskriva intressenter till systemet som skall utvecklas. Intressenter är alla de aktörer som har någon typ av intresse i systemet, vid utveckling och under drift. En central grupp intressenter är användarna, det vill säga de som ska nyttja systemet; men omfattar även de intressenter som på något sätt kommer att ha inflytande på eller påverkas av systemet. Exempel på de senare är de som ska finansiera systemet, de som ska administrera systemet eller aktörer som inte kommer att använda systemet men vars arbete ändå kommer att påverkas av det (t ex genom att deras arbetsuppgifter kommer att minska). Deltagande design (Schuler & Namioka, 1993), Agile utvecklig (Larman, 2003) och Total Quality Mangement (TQM) (Bergman & Klefsjö, 1994; Arthur, 1992) är exempel på ansatser som betonare vikten av intressenter på något sätt representeras och involveras i utvecklingen. Andra ansatser såsom Quality Function Deployment (QFD) betonar vikten av noggrann intressentanalys som grund för utvecklingen utan att direkt involvera dessa i själva arbetet (Mizuno & Akao, 1994).

Att identifiera intressenter är en omfattande uppgift som kräver djup förståelse av den kontext i vilket system skall användas. Metodmässigt kan identifieringen ske på ett flertal olika sätt. Ett sätt är att skapa en projektgrupp bestående av systemutvecklare och nyckelpersoner från den organisation i vilken systemet skall användas. Via diskussioner i projektgruppen kan ytterligare intressenter identifieras och sedan via samtal med dessa ytterligare intressenter tills samtliga relevanta aktörer täckts in. Det är lämpligt att beskriva intressenterna med korta beskrivningar av deras roll i organisationen, relation till det system som skall användas och vilket viktning deras synpunkter på system skall ges. Intressenter kan modelleras med Unified Modelling Language (UML) (OMG, 2001) samt såsom beskrivs av Hallberg (1999) genom en modell för att identifiera, strukturerna och prioriterar intressenter. Intressentanalyser är en central del i att ta fram kommunikationsplaner.

Exempel på relaterade behov: [2:1:1], [2:4:3]

7.2 Behovsanalys

Behovsanalyser är en viktig del i systemutveckling, men negligeras ofta eller ses som en implicit del av verksamhetsanalysen eller kravhanteringen (Davis, 2005). Många skiljer inte heller på begreppen *behov* och *krav*, och lite slarvigt kallas ibland behov för användarkrav respektive verksamhetskrav. Även om det i utvecklingsarbete kan vara lätt att blanda ihop vad som är behov och vad som är krav, så har begreppen två helt skilda betydelser. *Krav ställs på det som skall utvecklas, medan behov finns hos dem som skall nyttja produkten*. Behovsanalyser skall föregå kravspecificeringen och syftar till fånga intressenterna och verksamheternas verkliga behov. Detta skall göras innan design och beslut av tekniska lösningar påbörjas. Att genomföra behovsanalyser är ett omfattande arbete som ofta ges alldeles för lite utrymme i utvecklingsprojekt (Young, 2001, Boehm & Papaccio, 1998; Collin, 2003). Det är dock brister i denna del av utvecklingen som är mest kostsamma att korrigera i efterhand.

Behovsanalyser kan genomföras på ett flertal olika sätt (Hallberg, Andersson, Westerdahl, 2005; Hallberg, Ölvander, & Törne, 2006; Hallberg, Pilemalm, & Timpka, 1998; Arthur, 1992). Initialt handlar det om att fånga de identifierade intressenternas utsagor om verksamheten. Detta kan inkludera deras upplevda problem, behov eller önskemål om förändringar. Utsagorna kan samlas in på en rad sätt, genom intervjuer, enkäter, workshops, deltagande observation och genom att arbeta med prototyper och scenarier tillsammans med intressenterna (Hallberg, Ölvander, Johansson, & Törne, 2006; Alexander & Maiden, 2004). Utsagor uttrycker behov men ofta endast implicit, med ett innehåll av design,

teknik och förutfattade meningar. Därför är det viktigt att analysera utsagorna för att erhålla de verkliga behoven. Under denna process bör de formuleras på liknande sätt. Målet är att erhålla en behovsmängd som så bra som möjligt och som beskriver behoven av ett nytt system. Det är viktigt att också få bort ekvivalenta behov, det vill säga behov som kan ha formulerats något annorlunda men som i grunden har samman betydelse. Det är också viktigt att se om det finns några luckor i behovsmängden, det vill säga behov som missats i analysen och komplettera med dessa. Vidare är det viktigt att prioritera behoven med avseende på kundnytta. Behoven kan även kategoriseras i övergripande och elementära kategorier. Tillgång till användarrepresentanter under behovsanalyser är ett sätt att effektivt kvalitetssäkra resultatet. Slutligen bör behoven dokumenteras i form av en behovsspecifikation vilken tydligt beskriver de identifierade behov som anses tillräckligt viktiga för att ligga tillgrund för den vidare utvecklingen. Behovsspecifikationen utgör indata till kravspecificeringen. Ett sätt att överbrygga steget mellan behov och krav är att nyttja användningsfall (Eng: Use Case) (Kulak & Guiney, 2000). Det är rekommendabelt att på ett strukturerat sätt lagra behoven för att erhålla spårbarhet mellan utsaga, behov och krav (Hull, Jackson & Dick, 2004).

Exempel på relaterade behov: [2:1:1], [2:1:2]

7.3 Kravhantering

Krav beskriver vad system skall uppfylla och indirekt vad system *inte* skall uppfylla (Sommerville & Sawyer, 1997). Kravhantering är en nyckel aktivitet i så gott som all systemutveckling och nämns ofta som den viktigaste för att nå ett lyckat resultat (Young, 2001). Att ta fram rätt krav för ett system är fundamentalt; är systemet baserat på ”fel” krav spelar det ingen roll hur tekniskt avancerat det är, det kommer fortfarande att ha ”fel” funktionalitet och egenskaper (Collin, 2003). Studier visar att bristfällig kravhantering är den vanligaste orsaken till att systemutvecklingsprojekt misslyckas (Young, 2001, Boehm & Papaccio, 1998; Collin, 2003).

Krav uttrycker vad som ska produceras. – vad systemet ska göra utan att säga hur (Siddiqi and Shekaran, 1996). Kraven riktar sig till systemutvecklarna. Samtliga krav ska vara korrekta, entydiga, förståeliga, verifierbara och spårbara. Kravmängden ska dessutom vara komplett, konsistent samt modifierbar.

Under kravhanteringen formuleras krav på systemet utifrån identifierade behov (Hallberg, 1999). När ingen behovsanalys genomförts, måste kraven identifieras från exempelvis kravställande dokument, genom interaktion med intressenter eller analys av verksamhetsmodeller. Att genomföra en behovsanalys

rekommenderas dock starkt. Krav skall formuleras på ett enhetligt sätt samt bör kategoriseras och prioriteras. Om en prioritering gjorts av behoven kan denna med fördel användas för att erhålla prioritering av kraven (Hallberg, 1999). Målet är en kravmängd där varje krav är korrekt formulerat, entydigt, förståeligt, komplett, konsistent, verifierbart, modifierbart, spårbart och prioriterat samt att kravmängden är detaljerad till en nivå lämplig för den fortsatta utvecklingen (Wiegers, 2003). Även kraven bör lagras i en databas och ges en relation till motsvarande behov.

Slutligen produceras en kravspecifikation som beskriver och specificerar de krav som skall ligga tillgrund för designen; med koppling till behoven de är baserade på (Davis, 2005; Hull, Jackson, & Dick, 2005). Det kan även vara lämpligt att en grupp av kravexperter granskar kravspecifikationen så att den uttrycker rätt krav på systemet och att varje krav är korrekt, entydigt, förståeligt, komplett, konsistent, verifierbart, modifierbart, spårbart och prioriterat (Wiegers, 2003). Dock är användarrepresentanter mindre lämpliga för detta.

Kravhanteringen bör bedrivas iterativt under hela utvecklingen (Larman, 2003). Det innebär att kravhantering pågår genom hela utvecklingen, att kraven successivt uppdateras, kompletteras och modifieras. En anledning är att det ibland är lättare att kravställa senare i systemutvecklingsprocessen, när en konkret design och prototyper av system finns. Att nyttja prototyper för att identifiera och verifiera krav är en relativt vanlig ansats.

Exempel på relaterade behov:[2:1:1-2:1:7]

7.4 Design

Design innebär att beskriva hur systemet ska se ut och fungera. Medan kravspecificeringen syftar till *vad* som ska utvecklas anger designen *hur* detta ska realiseras, det vill säga tar fram lösningar som motsvarar kraven. Ofta tas en designspecifikation fram som beskriver hur funktioner och egenskaper skall realiseras i systemet.

Designen beskriver både synliga och osynliga aspekter av systemet som utvecklas – både funktioner och gränssnitt. Funktioner uttrycks t ex som systemkomponenter och moduler. Systemarkitektur är också en del av designen då den beskriver hur komponenter och moduler skall interagera. Rent konkret innebär design att man som systemutvecklare måste göra en rad avvägningar. Man måste t ex väga kostnad mot prestanda och beakta aspekter av stabilitet och IT-säkerhet när man jämför för- och nackdelar med olika designlösningar.

Designen är därmed en modell av systemet som utvecklas. Ofta tas flera modeller fram och visualiseras som systemskisser eller prototyper för att underlätta kommunikationen av designen med användare.

7.5 Realisering

Realisering innebär att omsätta designen till ett fungerande system. Detta kan innebära att systemet införskaffas, systemkomponenter införskaffats och konfigureras till ett fungerade system eller att system byggs från grunden. Ett ytterligare alternativ är att visa komponenter införskaffas och andra byggs från grunden. Att genomföra detta är inte trivialt, men helt beroende på vilken typ av system som skall realiseras. Exempelvis programvaror programmeras, ny kompetenser erhålls via utbildning eller rekrytering samt hårdvara sätt samman.

7.6 Verifiering

Verifiering innebär att värdera om en design alternativt ett system motsvarar kraven. En verifikation är en kontroll av att systemet uppfyller de krav som står i kravspecifikationen. Verifiering utförs därmed företrädesvis av systemutvecklare och blir i praktiken en jämförelse av kraven i kravspecifikationen och de funktioner det färdiga systemet har. När diskrepansen mellan design/system och krav är för stor måste utvecklingen itereras. Det vill säga designen/systemet måste förändra så att den/det bättre motsvarar kraven.

Exempel på relaterade behov:[2:6:1]

7.7 Validering

Validering innebär att krav, design eller system värderas mot behoven. Det vill säga en värdering av dessa för att bedöma om de ger önskad effekt för intressenterna. En bra grund för att validera är att säkerställa att behoven som identifierats är de rätta, vilket kan vara svårt att säkert avgöra. Valideringen av systemet bör ske i den verksamhet som systemet ska stödja. Man studerar helt enkelt prototyper av systemet eller det faktiska systemet i bruk för att avgöra hur användarna uppfattar detta och hur det fungerar i verksamheten. Rent konkret kan detta ske genom deltagande observation, intervjuer eller enkäter. Om valideringen visar att systemet inte lever upp till behoven bör måste det fastställas varför och avgöras vilka förändringar som behövs och ta med sig dessa till nästa iteration av utvecklingen. Att använda prototyper är ett sätt att validera krav och design tidigt i systemutvecklingen. Användare kan prova dessa för att avgöra om de motsvarar deras behov. På så sätt kan kostsamma omarbetningar

undvikas, och risken för ett färdigt system som inte motsvarar användarnas behov minskar.

Exempel på relaterade behov:[2:6:1]

8 Utvecklingsmetoder

I detta kapitel presenteras ett antal metoder som kan användas inom systemutveckling. Begreppet metod skall här beaktas i sin vidaste mening. Det inledande metoder är mer avgränsade, medan längre fram i kapitlet presenteras allt mer omfattande metoder. De beskrivna metoderna ska ses som goda exempel som kan tillämpas var och en för sig, i olika kombinationer och på olika sätt, beroende på projektets syfte och kontext.

8.1 Dokumentanalyser

Dokumentanalyser betecknar strukturerade studier av dokument. Detta används typiskt i den initiala utvecklingen, exempelvis för att studera en verksamhets struktur, verksamhet, vision, mål, aktörer och aktiviteter samt för att identifiera utsagor som grund för behovsanalyser. Dokumentanalyser av officiella måldokument som ställer krav på tolkning eftersom dokumenten ofta har konstruerats i en viss kontext och med ett visst syfte. Dessa dokument speglar inte alltid organisationers faktiska aktiviteter och mål. Liksom andra datainsamlingsmetoder, bör dokumentanalyser kombineras med andra metoder, för att en mer heltäckande bild ska nås. Dokumentanalyser kan användas både vid verksamhetsbeskrivning och för att finna behov och förändringar, exempelvis i måldokument, och vid identifiering av krav på IT-stöd.

8.2 Deltagande observation

Deltagande observation är observation av människor som agerar i deras naturliga miljö (Hasu & Engeström, 2000). Metoden härstammar från antropologi och etnografi men används idag inom systemutveckling. Här handlar det handlar om att studera användare i deras reella kontext, för att kunna utveckla system som är anpassade för denna. Metoden har visat sig var effektiv då det gäller förändringsarbete och teknikstödd förändring av arbetsuppgifter. Detta för att ge förstahandsinformation om de aktiviteter tekniken ska stödja; ofta med fokus på så kallade ”breakdowns” i aktiviteter, arbetsflöden eller -uppgifter, det vill säga när något skapar en störning i arbetsflödet (Hasu & Engeström, 2000). Lösningar kan sedan identifieras för att söka reducera dessa ”breakdowns”. Mer konkret kan utvecklaren sitta med användare när de utför sina arbetsuppgifter. Att be användaren ”tänka högt” är ett bra sätt att komplettera den information man får genom att endast studera arbetet; likaså kan videoinspelningar av hur arbetsuppgifterna utförs vara lämpliga.

Exempel på relaterade behov:[5:1:4]

8.3 Intervjuer

Intervjuer används för att samla information om och förstå människors erfarenheter och upplevelser; hur de konstruerar sin verklighet. De når en snävare grupp av individer än exempelvis enkäter och kan vara lämpliga när forskaren/utvecklaren vill fånga exempelvis enskilda aktörers perspektiv (Bernard, 2000) Inom systemutveckling kan intervjuer användas för att erhålla data för verksamhetsbeskrivning eller systemspecificering, om verksamhet, mål, visioner, aktiviteter, arbetsuppgifter och förändringsbehov och krav på lösningar. Men intervjuer är också användbara under realiseringen, valideringen och verifieringen som en källa till information huruvida systemet lever upp till fastställda krav och, om inte, hur det bör modifieras. Intervjuer kan exempelvis ingå som ett element i prototyputvärderingar. Intervjuer kan genomföras med enstaka individer eller grupper.

Exempel på relaterade behov:[2:1:2], [5:1:3], [5:1:5], [5:1:8]

8.4 The Critical Incident Technique

Enkäter kan användas för att nå information om användares arbetssituation, arbetsuppgifter, behov och krav på ett informationssystem. De relaterar därmed främst till utvecklingsfaserna verksamhetsmodellering och kravhantering och kan även användas för validering och verifiering av ett system. En typ av enkät som är effektiv då förändringsaspekter bör beaktas bygger på tekniken The Critical Incident Technique (CIT).

CIT bygger på användares självrapportering av incidenter som är kritiska i deras arbete, både incidenter som leder till tillfredsställelse och incidenter som leder till upplevda problem. Den information kan användas som grund för att genomföra förbättringar (Flanagan, 1954). Även enkäter och intervjuer kan användas för att samla in informationen från observatörerna. CIT har använts inom systemutveckling för att fånga användarnas upplevda problem i sitt dagliga arbete, behov av och krav på ett informationssystem (Hallberg, 1999). Enkäter och intervjuer ska konstrueras så att användarna får rapportera upplevda problem, lösningar på och åtgärder som skulle kunna lösa deras problem. Frågor som kan ställas inkluderar (Hallberg, 1999):

1. Vad var det senaste problemet du upplevde i din arbetssituation?
2. Vilka konsekvenser fick problemet för dig?
3. Hur löste du problemet?

4. Hur ofta inträffar problemet?
5. Kan du se någon åtgärd som skulle kunna reducera eller eliminera problemet i framtiden?

Exempel på relaterade behov:[2:1:2], [5:1:4], [5:1:5]

8.5 Future Workshops

Future Workshops härstammar från stadsplanering men som har vidareutvecklats inom systemutvecklingsansatsen Deltagande design (Eng: Participatory Design) (Kensing & Halskov Madsen, 1993). Future Workshops syftar till att aktivt involvera användare i systemutvecklingsprocesser genom att låta dem själva reflektera över sin situation och eventuella problem i relation till den, förändringsbehov i verksamheten och behov av informationstekniskt stöd för förändringar. Detta sker genom de tre faserna, *kritikfas, fantasifas* och *implementationsfas* där användare först beskriver den egna arbetssituationen och dess problem, sedan genererar futuristiska lösningar till problemen och i nästa steg omvandlar dessa till tekniskt och organisatoriskt implementerbara lösningar (Jungk, & Müllert, 1987). Arbetsgång kan vara:

1. Reflektera individuellt eller i mindre grupp över egen arbetssituation. Vilka problem upplevs?
2. Generera lösningar till identifierade problem. Dessa är i detta läge visionära och styrs ej av tekniska, ekonomiska eller organisatoriska restriktioner.
3. Omvandla dessa lösningar till realiserbara lösningar.
4. Arbeta i större grupp med att sammanställa och kategorisera lösningar.

Syftet med Future Workshops, förutom aktiv användarmedverkan, är att undvika en alltför ensidig fokusering på tekniken från början och istället nå användares verkliga behov. Dessutom är principerna för en Future Workshop lätt att lära för användare utan systemutvecklingskunskap och kan utföras med enkla hjälpmedel; papper, penna, whiteboard och post-it lappar räcker ofta.

Exempel på relaterade behov:[2:1:2], [5:1:3], [5:1:5], [5:1:6], [5:1:9]

8.6 Designtekniker för användarmedverkan

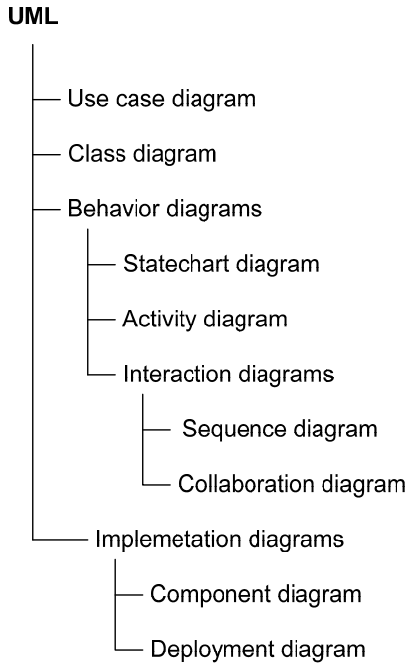
Det finns ytterligare en rad tekniker som utvecklats speciellt för att stödja aktiv användarmedverkan, samarbete mellan systemutvecklare och användarrepresentanter. Hit hör exempelvis arbete med organisationskartor, mock-ups (pappersmodeller av organisation eller system), dramatiseringar av arbetssituationer och arbetsuppgifter och bruk av metaforer för att beskriva verksamhet eller informationssystem (Ehn, Davies, Brattgård, Hägerfors, Nilson,

Dalholm, & Mitchell, 1996). Gemensamt är att teknikerna ska vara lätta att lära, ställa låga krav på förkunskaper i systemutveckling och kunna utföras med enkla hjälpmedel såsom t ex papper, sax, penna. PICTIVE är exempelvis en verktygslåda bestående av olika grafiska element (i pappersformat) som används för att användare ska kunna medverka vid gränssnittsdesign (Muller, 1995).

Exempel på relaterade behov:[5:1:3], [2:4:2]

8.7 Unified Modeling Language

Unified Modeling Language (UML) är ett standardiserat modelleringsspråk för visualisering, specificering, konstruktion och dokumentation (OMG, 1999). Ursprungligen utvecklades UML för mjukvaruintensiva system, men har visat sig användbart för att beskriva de flesta typer av system. Språket inkluderar en rad notationer och diagram för att beskriva såväl dynamik som statiska egenskaper hos system (Figur 2). UML innehåller möjligheten att skapa egna tillägg till notationerna vilket möjliggör anpassning av språkets syntax och semantik.

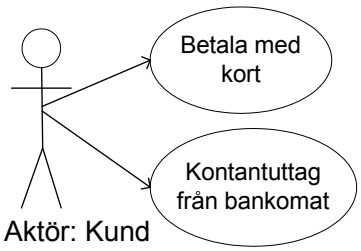


Figur 2. Notationer som ingår i UML

8.8 Användningsfall

Användningsfall beskriver hur system används alternativt är tänkta att använda. Det består av en text som beskriver själva användningen, samt diagram som illustrerar aktörers interaktion med systemet (Figur 3). Att utnyttja användningsfall som en första övergripande beskrivning av vilken funktionalitet ett system skall tillhandahålla är populärt (Kulak & Guiney, 2000). Detta eftersom de både på ett enkelt och tekniskt sätt beskriver de övergripande funktionella kraven.

Utveckling med användningsfall skall utgå från de mest centrala funktionerna som system skall tillhandahålla och de aktörer som skall nyttja dessa funktioner. Funktionerna bryts sedan ner i mer specifika funktioner. Baserat på användningsfallen kan mer konkreta krav formuleras på systemen. En god hjälp med att formulera användningsfall är att nyttja någon av de mallar som finns tillgängliga i exempelvis RUP (Kruchten, 2004).



Figur 3 Exempel på användningsfallsdiagram

Exempel på relaterade behov:[2:2:2]

8.8.1 Mall för användningsfall.

Nedanstående punktlista representerar en mall för beskrivning av användningsfall (Hallberg, Ölvander, & Törne, 2006). För varje användningsfall skall följande beskrivas:

- Identifierare; ett unikt nummer för varje användningsfall som anges av den som administrerar användningsfallen i projektet. Inom parantes anges versionen av användningsfallet.
- Namn; användningsfallet namnges som ett verb som beskriver vad aktörerna gör med systemet.
- Kortfattad beskrivning; den interaktion mellan aktören och systemet som skapar det för aktören observerbara resultatet.

- Nytta; beskriver den nytta som erhålls vid användningen av systemet i enlighet med användningsfallet.
- Behov som motiverar; en lista på de behov som motiverar användningsfallet.
- Primära aktörer; beskriver de primära aktörer som nyttjar systemet enligt användningsfallet, i subjektsform och gärna med namn som beskriver aktörernas roller. Primära aktörer är de som initierat användningsfallet. De kan vara såväl mänskliga som tekniska.
- Sekundära aktörer; beskriver de sekundära aktörer som har nytta av systemet enligt användningsfallet, i subjektsform och gärna med namn som beskriver deras roll. Sekundära aktörer initiera inte användningsfallet utan kan helt passivt ha nytta av resultatet. De kan vara såväl mänskliga som tekniska.
- Stödjande aktörer; de aktörer som används av systemet. De är inte en del av systemet och kan därför inte designas eller anpassas. De består oftast av externa system eller personer.
- Användningsfrekvens; en uppskattning av hur ofta systemet kommer att användas i enlighet med användningsfallet. Om information saknas lämnas fältet tomt.
- Förutsättningar; beskriver vad som måste vara uppfyllt för att systemet skall kunna användas i enlighet med användningsfallet.
- Funktionella krav; beskriver de funktionella krav som användningsfallet ger upphov till, med ”skall”-formulering.
- Icke-funktionella krav; beskriver de icke-funktionella krav som användningsfallet ger upphov till, med ”skall”-formulering.
- Ingår i användningsfall; är en referens till de användningsfall som detta användningsfall ingår i.
- Omfattar användningsfall; ger en referens till de användningsfall som ingår i detta användningsfall.
- Författare; namnet på den som formulerat användningsfallet.
- Datum; en lista med datum då användningsfallet formulerats och uppdaterats. Vid uppdateringar anges datum samt namnet på den som genomförde uppdateringen inom parantes. Även annan information om uppdateringen kan anges.
- Exempel på relaterade behov:[2:1:2], [5:1:3], [5:1:5], [5:1:8]

8.9 Prototyping

En prototyp är en modell av ett system. Dessa kan användas för att ge användare en möjlighet att påverka och modifiera systemet och att successivt identifiera och eliminera designproblem för att nå en slutlig bättre systemlösning (Fransson, 1999). Dessutom kan prototyper användas, tidigt i systemutvecklingen, för att

identifiera krav på verksamhet och ledningssystem. Prototyper stöder en iterativ systemutveckling och aktiv användarmedverkan för att öka det färdiga systemets användbarhet.

Prototyputveckling kan vara baserad på en mängd olika ansatser som kan användas för olika ändamål. Exempelvis skiljs på explorativ och evolutionär prototyputveckling. Den förra innebär att snabbt ta fram en rad modeller av systemet för att identifiera användarnas behov. Den andra fokuserar på gradvis utveckling och modifiering av en modell som kan användas som utgångspunkt för det slutliga systemet. Vidare kan prototyper vara horisontella eller vertikala. Med horisontella prototyper kan en god överblick ges av systemet men de kommer att ha begräsad funktionalitet. Med vertikala prototyper implementeras vissa begränsade delar av systemet med hög funktionalitet (Avison & Fitzgerld, 1995). Prototyper kan därmed utgöras av alltifrån pappersmodeller och Powerpointpresentationer till sammanlänkade webbsidor men kan också bestå av olika mjukvaruapplikationer. En annan uppdelning kan göras mellan prototyper som fokuserar på funktionalitet och de som fokuserar på gränssnitt, även om de två egentligen interagerar med varandra och inte kan separeras. De fördelar prototyparbete ger innefattar (Nielsen, 1993):

- Högre kvalité på färdiga system, på grund av mer korrekta krav har kunnat identifieras, vilket resulterar i högre grad av användbarhet av det slutliga systemet.
- Billigare system då användare tidigt kan identifiera bl a överflödiga funktioner.
- Högre acceptans och förståelse för systemet från användarna.
- Stora möjligheter för användarna att påverka systems utforming under utveckling.

Utvecklandet av prototyper är ett vedertaget inslag i många systemutvecklingsansatser. I användarcentrerad systemutveckling ges prototyparbete generellt stort utrymme. Det är viktigt att finna en balans mellan prototyper som fokuserar på systeminnehåll och funktionalitet och arbete som fokuserar enbart på gränssnittsfrågor. Generellt bör systeminnehåll fokuseras tidigt i processen för att säkerställa att systemet svarar mot användarnas grundläggande krav och behov av information. Explicita gränssnittsfrågor bör ges ökat utrymmer ju längre fram i utvecklingsprocessen man kommer.

Exempel på relaterade behov: [5:1:3]

8.9.1 Utvärdering av prototyper

För att prototyperna och därmed systemet under utveckling ska kunna förbättras krävs utvärdering av dem. Det finns en rad tekniker för utvärdering av prototyper. Den som är mest i linje med användarcentrerad, iterativ systemutveckling är scenariobaserad utvärdering. Scenarier är beskrivningar av, konstruerade användarsituationer med avseende på arbetsuppgifter som ska utföras med hjälp av systemet; en beskrivning av en aktivitet definierad som en rad, ofta sekventiella, handlingar. Oftast är de beskrivna i text form (Carroll, 2000). Scenarier kan vara beskrivna på en övergripande nivå som en huvuduppgift med ett antal underuppgifter som är mer detaljerat och formaliserat (Hallberg, Ölvander, Johansson, & Törne, 2006).

I scenariobaserade utvärderingar interagerar användarrepresentanter med systemet och utfallet kan mätas både kvantitativt, med avseende på exempelvis felfrekvens, och kvalitativt, med avseende på subjektiva upplevelser av systemet. Även om användare alltid bör medverka vid utvärdering av system under utveckling finns även andra typer av prototyputvärderingar. I expertutvärderingar utvärderar ett antal experter systemet, exempelvis utifrån projektets designriktlinjer eller rådande heuristiker (allmänna krav på användbarhet). Oftast sker sådana utvärderingar tidigt i designfasen för att eliminera grundläggande fel och brister i systemet. Grupputvärderingar genomförs gemensamt av användarrepresentanter och systemutvecklare och kan även innefatta andra experter såsom t ex ergonomer.

8.10 Service oriented architechture

Tjänsteorienterad arkitektur (Service Oriented Architechture, SOA) ger möjligheten att utveckla distribuerade system, för i förstahand IT-system, som organiseras som en struktur för kommunikation mellan system genom kommunicerande tjänster (McGovern, Ambler, Stevens, Linn, Jo, & Sharan, 2003). En tjänst i SOA avser en betjänande funktion som är väldefinierad, självständig och oberoende av sin omgivning. I system uppbyggda enligt SOA är resurser tillgängliga för andra system inom ett nätverk som oberoende tjänster, som hittas, bedöms, avropas på ett standardiserat sätt. SOA möjliggör återanvändning av befintliga system i form av tjänster genom att dessa inkapslas i någon lämplig kontext. SOA är inte bara avsett för mjukvarubaserade system, men det är i dessa som SOA fått sin främsta användning. Webtjänster är en möjlighet att tillägna sig SOAs priciper, det vill säga en tjänste- och komponentorienterad ansats för att ge tillgång till information genom kraftfull funktionalitet.

Följande arkitekturprinciper gäller för SOA

- Inkapsling – att befintliga system kan kapslas in, genom ett SOA gränssnitt.
- Lösa kopplingar – tjänster skall ha lösa koppling av beroenden, helst bara genom kännedom om varandra.
- Kontrakt – tjänster måste uppfylla det som står i dess beskrivning. Det vill säga de som nyttjar tjänsterna måste kunna lita på att de uppflylls på det sätt som är beskrivet.
- Abstraktion – Förutom den beskrivning som gör av tjänstens funktion, gränssnitt och beteende skall tjänstens logik vara dold.
- Återanvändbarhet – tjänster skall kunna återanvändas för att skapa ny funktionalitet och nya system.
- Autonomi – Tjänster skall inte behöva mer information än den som anges som indata för att kunna utföra det som utlovats.
- Optimering – Det skall baserat på tjänstebeskrivningarna vara möjligt att identifiera den tjänsterealisering som passar bäst i varje givet ögonblick.
- Upptäckbarhet– Tjänster måste vara utformade så att deras externt tillgängliga beskrivning gör att de kan upptäckas och bedömas .

För att skapa system baserade på SOA kan en särskild modellerings- och designmetodik användas, Service Oriented Analysis and Design. En sådan ansats baseras ofta på underliggande tekniker som xml, http, SOAP, WSDL, BPEL och UDDI.

- Simple Object Access Protocol (SOAP) är ett meddelande baserat protokoll för att nyttja tjänster; baserat på XML.
- Universal Description, Discovery and Integration (UDDI) utgör ett register (en katalog) över tillgängliga tjänster. UDDI är utvecklat för att möjliggöra automatisk upptäckt av tillängliga webtjänster. Med en UDDI-browser kan även människor läsa och ta del av informationen i UDDI baserade register.
- Web Services Description Language (WSDL) är en standardiserad metod för att beskriva vilka funktioner som finns tillgängliga genom en viss webtjänst, och vilka variabler som måste överföras vid anropa av tjänster.
- Business Process Execution Language (BPEL) är ett språk som beskriver hur tjänster kan aggregeras för att skapa nya tjänster utifrån existerande tjänster. Språket kan användas till att beskriva interaktioner mellan webbtjänster.

SOA baserade lösningar behöver dock inte nödvändigtvis bygga på några eller någon av dessa tekniker. SOA måste i första hand ses som en tankemodell, ett koncept, ett sätt att designa hur olika applikationer skall fungera och utbyta information.

Begrepp som är relaterade till SOA och Web Services är (Benatallah, Sheng, & Dumas, 2003):

- Elementary services är tjänster som ger åtkomst av viss funktionalitet, utan att nyttja andra tjänster. Dessa tjänster kallas ibland även för Basic services (Rabi & Benatallah, 2002).
- Composite services är tjänster som är sammansatta av elementary services och andra composite services. Dessa tjänster kallas ibland även för Integrated services. (Rabi & Benatallah, 2002)
- Service container är tjänster som innehåller ett flertal liknade tjänster. Ger dynamik då tjänsteproducenterna kan välja vilken tjänsterealisering som skall nyttjas för att möta kundbehoven.
- Konsument - använder tjänster och erhåller en prestation genom det gränssnitt som tjänsten tillhandahåller. Från konsumentens perspektiv är tjänsten en fasad mot producenten. Konsumenten behöver inte ha kännedom om vilka producenter som tillhandhåller olika tjänster, utan kan söka efter tjänsterna via en katalogtjänst. Genom katalogtjänsten kan konsumenten få reda på vad en tjänst tillhandahåller och hur åtkomsten kan tillgodoses. Men det är inte något som hindrar att information om producentens och konsumentens identiteter utväxlas som en del av den information som utbyts. Om detta är önskvärt så publiceras även producentens identitet vid publicering av en tjänst.
- Producent - producerar tjänster och är normalt inte medveten om vilka konsumenterna är och hur de kommer att använda tjänsterna. Att en konsument inte är kopplad till en specifik producent innebär möjlighet till evolutionär systemutveckling. En tjänsteproducent kan enkelt bytas ut så länge den nya producenten tillhandahåller samma tjänst som tidigare. En viktig princip är att producenten alltid ska exportera *all* tillgänglig information genom gränssnittet mot konsumenten, eftersom behovet av information för en okänd konsument inte kan förutses i sin helhet.

8.11 Quality Function Deployment

Quality Function Deployment (QFD) (Mizuno, S. & Akao, 1994) är ett kvalitetssystem med ursprung i japanskt kvalitetstänkande, och är nära besläktat med Total Quality Managment (TQM) (Bergman & Klefsjö, 1994; Arthur, 1992). QFD har spridits till en rad olika applikationsområden, men är mest känd för sin användning inom produktutveckling, främst bilindustrin (Akao, 1997). QFD har dock även använts vid tjänste- och systemutveckling (Hallberg, 1999). Inom systemutveckling är QFD främst lämpat för att stödja identifieringen av användarnas behov och omvandlingen av dessa behov till krav på systemet, och

därefter till bestämning av designattribut (funktioner, egenskaper mm). QFD består av en filosofi, ett antal kvalitetsverktyg och appliceringsmodeller (Hallberg, 1999). Fundamentet i filosofin är att fokusera utvecklingen på de aspekter av en produkt/tjänst som ger värde för användaren. QFD innehåller en rad grafiska verktyg som stöder detta, exempelvis relationsdiagram, hierarkiska diagram, tabeller och matriser (Mizuno, 1988). Vid systemutveckling kan en tabell som kallas kundrösttabell användas för identifieringen av användares behov, från exempelvis deras egna beskrivningar av sin arbetssituation och önskemål av egenskaper hos ett framtida informationssystem. Matriser kan användas för att omvandla behov till krav på systemet och sedan till designattribut. Detta genom att införa samband för hur väl ett tekniskt krav motsvarar de identifierade behoven, och därefter kan sambanden mellan krav och designattribut identifieras. Även prioriteringar av hur viktiga olika behov anses vara att uppfylla, kan överföras till prioriteringar av kraven, och därefter till designattribut. Det mest kända verktyget i QFD är den matris som kallas Kvalitetshuset (Hauser & Clausing, 1988). QFD kan därmed stödja systemutvecklingsprocessen vid verksamhetsbeskrivning, systemspecificering och design, främst genom att identifiera behov och krav på systemen, att prioritera dessa samt att säkra spårbarhet mellan dem. QFD kan även användas vid riskhantering för att identifiera, prioritera och etablera relationer mellan risker.

Exempel på relaterade behov:[2:1:2], [6:1], [6:2], [6:3], [6:4]

8.11.1 Kundrösttabell

Kundrösttabell (eng: Voices of Customers Table, VCT) är ett av många kvalitetsverktyg som utvecklats inom QFD (Mazur, 1992). I kundrösttabellen skrivs kundens önskningar in som utsagor. Dessa utsagor analyseras därefter utifrån (1) *vem* som anses behöva dem, (2) *vad* de anser sig behöva, (3) *när* de behöver det, (4) *var* det behöver användas, (5) *varför* behovet finns och (6) *hur* de vill att det skall fungera. Analys med stöd av kundrösttabell har som mål att kunna formulera faktiskt behov. Kundrösttabellens primära funktion är att underlätta analysen av vad kunden egentligen behöver.

8.12 Användarmedverkan

De flesta systemutvecklingsansatser talar idag om vikten av användarmedverkan i systemutveckling. Motivet är oftast att det är användarna som känner sin domän bäst och som vet vilket behov av stöd de har. Det finns dock många olika sätt att involvera användarna, till olika grad samt att ge dem olika mandat för att

påverka. Användarna kan ses som konsulter som kallas i vid behov under utvecklingen för att ge sin syn på det blivande systemet. De kan användas exempelvis för att validera verksamhetsmodeller eller utvärdera prototyper av systemet, eller för att utvärdera systemet i utvärderingsfasen. Att vänta med att involvera användarna tills utvärderingen av ett nästan färdigt system är dock inte att rekommendera, då utvecklarna kan få många obehagliga överraskningar som kan vara kostsamma att rätta till. De systemutvecklingsansatser som betonar vikten av användarmedverkan hävdar istället att användarna bör involveras *aktivt* genom *hela* systemutvecklingsprocessen. Det sker ofta genom så kallade designgrupper bestående av systemutvecklare och användarrepresentanter som tillsammans arbetar genom alla utvecklingsfaserna. Ansatser som betonar aktiv användarmedverkan är exempelvis Deltagande design and Joint Application Design (JAD) (Schuler & Namioka, 1993; Carmel, George, & Nunamaker, 1992).

Det finns en rad tekniker/metoder för att möjliggöra användarmedverkan och fånga användarnas behov och perspektiv på systemet. Exempel är direkt konsultation, intervjuer, enkäter, scenarier och prototyputvärderingar. Ansatser som Deltagande design och JAD har dessutom tekniker som är direkt anpassade för att stödja användarmedverkan, exempelvis Future Workshops (där användarna får identifiera behov i sin nuvarande arbetssituation och omvandla dessa till lösningar) och PICTIVE (där användare får designa sina egna gränssnitt) (Muller, 1993). Användarmedverkan är mycket viktigt i behovsanalysen och designprocessen, medan i kravhanteringen och realiseringen är deras medverkan mindre viktig.

Respondenterna nämnde hur viktigt det är att på rätt sätt involvera användare vid utveckling av ledningssystem. Användarna bör vara motiverade, representativa och heterogena (d v s olika typer av användare bör involveras). De bör involveras på ett tidigt stadium och fortsätta att aktivt och kontinuerligt medverka genom hela systemutvecklingsprocessen. Respondenterna talar även om hur viktigt det är med kompetens vid användarmedverkan för att kunna genomföra realistiska användarstudier. De nämner också att det idag är svårt att frigöra FM personal som kan agera användarrepresentanter. Detta gör att det är viktigt att vara effektiv när användarrepresentanter blir tillgängliga.

Exempel på relaterade behov: [5:1:1-5:1:11]

8.13 Arkitekturramverk

En metod för utveckling av systemarkitekturer är att nyttja ett arkitekturramverk för att erhålla sammanhängande och konsekventa modeller Dessa definierar vyer

över hur modellerna beskrivs och presenteras samt skapar relationer mellan vyerna. Nyare arkitekturramverk nyttjar ofta en metamodell som bindersamman de objekt som beskrivs och beskriver även hur de förhåller sig till varandra. Det finns många arkitekturramverk. Nedan beskrivs MODAF och TOGAF.

Exempel på relaterade behov:[3:1], [13:1], [1:8], [6:1], [6:2], [6:3], [6:4]

8.13.1 Ministry of defence architectural framework (MODAF)

Ministry of Defence Architectural Framework (MODAF) är ett arkitekturramverk framtaget för MOD (Ministry of Defence) i Storbritannien och används som styrning vid arbete med och framtagning av Enterprise Architecture för att påverka arkitekturarbetet så att det styrs i enlighet med gemensamma designbeslut. MODAF bidrar med en strukturerad beskrivning och en uppsättning enhetliga uttryck för beskrivning av komplexa arkitekturer. Syftet med ett arkitekturramverk som MODAF är att definiera de komponenter som ingår i en arkitektur, till exempel organisationer, processer och informationsflöden samt dess relationer till varandra. MODAF kan även beskriva till exempel systemarkitekturer där gränssnitt, dataspecifikationer och systemkomponenter ingår samt hur relationerna mellan dessa ser ut. Informationen som presenteras i ett arkitekturramverk som MODAF kallas vyer. Syftet med vyerna är att presentera system- eller verksamhetselement utifrån olika perspektiv. Vad perspektiven ska beskriva beror på vilka av systemets intressenter som informationen är avsedd för; olika intressenter har olika roller och därmed skiftande behov av information om system- eller verksamhetselementen (MODAF, 2008). Vyerna i MODAF är grupperade i sju perspektiv.

- **All Views (AV)** Ger summarisk information rörande arkitekturmodellen och tillhandahåller information som gör det möjligt att leta i och indexera modellen.
- **Strategic Views (StV)** Dokumenterar den strategiska bilden rörande förmågor och hur dessa utvecklas över tiden. Detta understödjer förmågeplanering och styrning.
- **Operational Views (OV)** Dokumenterar operationella processer, relationer samt deras omgivning för att möjliggöra operationell analys och kravutveckling.
- **System Views (SV)** Dokumenterar systems (resursers) funktionalitet och interkonnektivitet för att stödja systemanalys och livscykelhantering.
- **Technical Views (TV)** Dokumenterar policies, standarder och begränsningar som skall gälla för den övergripande arkitekturen.

- **Service Oriented Views (SOV)** Dokumenterar tjänsters funktionalitet, begränsningar och inbördes beroenden.
- **Acquistition Views (AcV)** Dokumenterar program, beroenden, tidsscheman och programstatus för att tillhandahålla styrinformation samt programsynkronisering.

8.13.2 TOGAF – The Open Group Architecture Framework

The Open Group Architecture Framework (TOGAF) är en omfattande ansats ,med syfte att utforma, planera, införa och styra vidareutvecklingen av en verksamhet informationsarkitektur. Denna arkturer beskrivs vanligen i fyra domäner: ”Business”, ”Application”, ”Data”, och ”Technology”. TOGAF är generell och möjliggör för användare att designa, utvärdera och bygga rätt slags arkitektur för deras organisation, samtidigt som de minskar kostnaderna för att planera, designa och implementera arkitekturer baserade på öppna systemlösningar. Nyckeln till TOGAF’s framgång är att det är en tillförlitlig och praktisk metod, TOGAF Architecture Development Method (ADM) används för att definiera företagsbehov och därefter utveckla en arkitektur som uppfyller dessa (The Open Group, 2006). The Open Groups vision med TOGAF och ADM är att erfarenhetsbaserad information ska kunna integreras med avseende på hur arbetet med EA ska gå tillväga. Detta har gett en generell metod, som i sin tur leder till särskilda hänvisningar för modellen, med andra relevanta arkitektoniska tillgångar (The Open Group, 2006).

8.14 IT-säkerhet

Säkerhetsfrågor är ofta svåra att hantera i systemutvecklingssammanhang. Ofta löses säkerhetsrelaterade frågor i realiseringsfasen eller, än värre, genom uppdateringar när systemet redan i drift. För att få en effektiv och flexibel säkerhetslösning måste behov av säkerhet beaktas redan i de tidigaste faserna av utvecklingen (Hallberg & Hallberg, 2006; Devanbu & Stubblebine, 2000).

Ett problem är att säkerhetsproblem ofta beskrivs med andra modeller än de som nyttjas för systemutveckling. Detta får som konsekvens att systemutvecklares och säkerhetsexperters modeller och därmed utveckling inte är kompatibla. För att få ett fullgott skydd, vilket står i proportion till de uppgifter systemet skall lösa och den miljö som det skall användas i, bör säkerhetsaspekterna tas upp redan i verksamhetsanalysen. Görs detta fångas behov upp på ett tidigt stadium varpå dessa behov kan omformuleras till tydliga krav vilka senare påverka design- och realiseringsaktiviteterna.

Modellbaserad utveckling förbättrar förutsättningarna för en tillförlitlig säkerhetslösning. Det är dock ingen garanti för ett godtagbart säkerhetsskydd. Säkerhetsfrågeställningarna måste fortfarande uppmärksammas i ett tidigt skede så att de harmoniserar med utvecklingen av funktionalitet och därmed, med de behov som verksamheten har. Med modellbaserad systemutveckling kan mer preciserade systembeskrivning erhållas vilket är en förutsättning för en adekvat riskuppskattning.

Exempel på relaterade behov:[3.1], [9.1], [9.2]

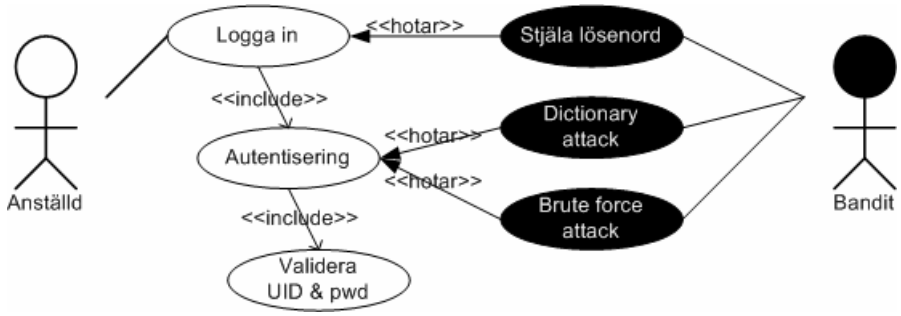
8.14.1 Modellering av säkerhet

Det finns flera ansatser för att modellbaserat beskriva säkerhetsbehov och -krav. En del av dessa ansatser är avsedda för att användas inom Rational Unified Process (RUP) (Kruchten, 2004).

8.14.1.1 Misuse Cases

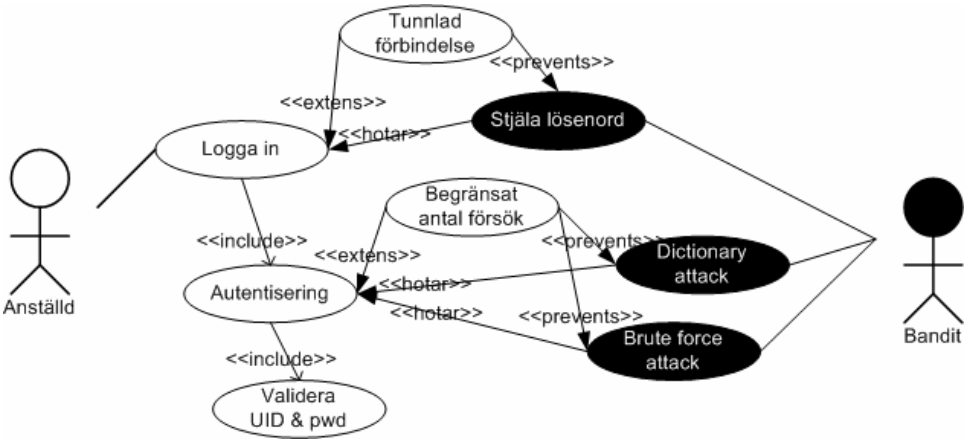
Ett användningsfall används för att beskriva hur ett system skall bete sig då det påverkas utifrån. Resultatet av ett användningsfall består av funktionella krav samt en beskrivning av vilka funktioner som påverkar vilka aktörer.

Säkerhetsbeskrivningar syftar till att motverka att system missbrukas. För att åstadkomma detta måste möjligheter till att utnyttja systemet felaktigt identifieras. Ett sätt att göra detta är att använda Misuse caseS vilket är en utveckling av traditionella användningsfalls (Sindre & Opdahl, 2000). Figur 4 visar ett exempel på ett enkelt Misuse Sase.



Figur 4. Misuse Case

Genom att visa på möjliga hot i samma modell som de funktionella kraven ges möjligheten att komplettera modellen med funktionella krav som kan reducera konsekvenserna av hoten. Figur 5 visar ett Misuse Case där åtgärder mot hoten har introducerats.



Figur 5. Misuse case med motåtgärder

Till modellen hör även en textbeskrivning av Misuse Case:et precis som i vanliga användningsfall.

Användningsfall och Misuse Cases kan användas i flera aktiviteter under utvecklingen – från verksamhetsmodellering till realisering av systemet. Dock är det viktigt att ett Misuse Case och användningsfall hålls på samma abstraktionsnivå så att de funktionella kraven och säkerhetskraven motsvarar varandra.

8.14.1.2 Viewpoints

Viewpoints är inte specifikt en metod för att utvinna säkerhetsegenskaper under systemutvecklingen (Sommerville, Sawyer, & Viller, 1998). Dock fungerar metoden för att identifiera såväl funktionella som icke-funktionella krav för hela systemet.

En viewpoint (vy) beskriver en uppgift ur en användares perspektiv. Ur vyn skall det framgå vem användaren är (roll) och vilken uppgift användaren löser. Kopplat till varje användare finns ett antal concerns (angelägenheter) som tillämpliga för användaren (Tabell 2 Tabell för att beskriva relationer mellan ”Viewpoint” och ”Concern”Tabell 2). Ett concern är vad som driver kravutvinningen från användarperspektivet.

Tabell 2 Tabell för att beskriva relationer mellan ”Viewpoint” och ”Concern”

	Concern_1	**Concern_2**	**Concern_3**
Viewpoint_1			
Viewpoint_2			
Viewpoint_3			

Ett concern är globalt i den betydelsen att den kan vara giltig för flera olika viewpoints i systemet.

8.14.1.3 UMLsec

Det räcker inte med att enbart identifiera hot och risker. Dessa måste följas upp och resultatet av genomförda åtgärder måste bli en del av systemet. För att få ett långsiktigt resultat bör säkerhetsfrågor uppmärksammas och hanteras på designnivå. UMLsec är en vidareutveckling av UML bland annat genom att ett antal stereotyper införts. En stereotyp är ett sätt att kapsla in gängse regler för säkerhetsmässig systemutveckling. Det är därmed ett sätt att föra in säkerhetsegenskaper redan i kravfasen, men också ett stöd under större delen av utvecklingsprocessen. UMLsec utvecklades för UML av Jürjens (2005). Det finns dock ingen uppdaterad version av UMLsec för UML2.

8.15 Kvalitetsdriven kravhantering

Kvalitetsdriven kravhantering är en metod för att identifiera krav som baseras på kvalitetsverktyg som finns beskrivna i QFD litteraturen och genomförs i sex steg: (1) Insamling av data, (2) Identifiering av utsagor, (3) Identifiering av behov, (4) Analys av behov, (5) Identifiering av krav och (6) Analys av krav (Hallberg, Andersson & Westerdahl, 2005).

8.15.1 Insamling av data

Under den första aktiviteten, Insamling av data, samlas information in som är av intresse för att kravställa det system som skall utvecklas. Insamling av data kan ske med ett flertal olika metoder och tekniker, exempelvis intervjuer, observationer, workshops och enkäter. Ett effektivt sätt att samla in relevant information är att efterfråga upplevda problem, något som indikerar situationer som upplevs otillfredsställande i något avseende. Otillfredsställande situationer avviker från det förväntade eller det önskade. Problem kan vara existerande eller tänkbara i framtida situationer. Utdata från detta steg skall vara i textformat.

8.15.2 Identifiering av utsagor

Under den andra aktiviteten, Identifiering av utsagor, analyseras texten för att identifiera utsagor. En utsaga är en text som innehåller information vilken sedan kan nyttjas för att bestämma behov. En utsaga skall inte vara för omfattande eller innehållsrik utan bör bestå av en eller ett par meningar, men ändå ge en förståelse för vad texten beskriver.

8.15.3 Identifiering av behov

Under den tredje aktiviteten, Identifiering av behov, genomförs en analys av utsagorna med syfte att identifiera behov. För detta ändamål används kundrösttabeller (Mazur, 1992). Varje utsaga kan innehålla ett eller flera behov, men kan också vara så innehållsfattig att inget behov alls kan extraheras. Resultatet av denna aktivitet utgörs främst av ett antal behovsformuleringar.

8.15.4 Analys av identifierade behov

Under den fjärde aktiviteten, Analys av identifierade behov, genomförs en analys av behovsformuleringarna. I ett första steg genomförs detta med relationsdiagram, där likartade behov grupperas så att kategorier skapas. Dessa kategorier grupperas i sin tur med liknande kategorier. Identifierade behov kan vara enkla eller sammansatta. Enkla behov kan inte delas upp ytterligare, medan sammansatta är möjliga att dela upp i flera olika behov. I det senare fallet kan behovet ses som en kategori innehållande andra kategorier och behov. Under analysen sorteras även dubbletter av behov bort och formuleringarna ensas.

Det andra steget i denna aktivitet inleds med att överföra relationsdiagrammen till hierarkiska diagram (Hallberg, 1999). Dessa diagram analyseras sedan vidare i syfte att upptäcka om behov som saknas, och som i så fall bör kompletteras.

8.15.5 Identifiering av krav

Under den femte aktiviteten, Identifiering av krav, analyseras behoven vidare för att identifiera de krav som skall ställas på det system som skall utvecklas. Kraven skall uppfylla svaret på frågan ”Hur kan detta behov tillfredsställas?” och formuleras på formen ”Systemet skall …” utan att specificera *hur* kravet skall realiseras.

8.15.6 Analys av identifierade krav

Under den sjätte aktiviteten, Analys av identifierade krav, genomförs en analys av de krav som har identifierats. Detta sker genom att använda samma typ av diagram och arbetssätt som under analysen av behov i den fjärde aktiviteten.

8.16 Rational Unified Process

Rational Unified Process (RUP) är ett processramverk som omfattar hela utvecklingscykeln för system, i första hand mjukvara (Kruchten, 2004). Den initiala utveckligen av RUP gjordes av personer med rötterna i en objektorienterad systemsyn. RUP stödjer evolutionär systemutveckling och är i hög grad modellbaserad. För modellering används modelleringsspråket Unified Modeling Language (UML). Ramverket är organiserat i två dimensioner. Den första dimensionen beskriver hur utvecklingsprojektet ska genomföras över tiden. Den andra dimensionen är uppdelad i de discipliner (arbetsuppgifter) som skall genomföras under projektet. Varje disciplin beskrivs som en mängd aktiviteter sammanlänkade i ett arbetsflöde. För varje aktivitet beskrivs vilka roller som skall medverka i genomförandet samt vilka modeller och dokument som skall produceras.

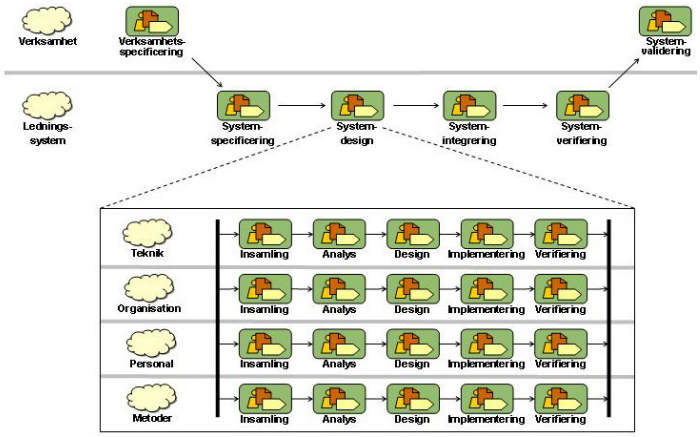
8.17 Verksamhetsutvecklingsmetoden för Ledningssystem

Verksamhetsutvecklingsmetoden för Ledningssystem (VUM-LS) bygger på de principer och ansatser från RUP, UML, användarcentrerad systemutveckling, principer om användbarhet och Quality Fuctionen Deploment. Metoden har utvecklats baseras på erfarenheter från utvecklingsprojekt inom FM gjorda av FOI. Målet med VUM-LS är att beskriva en beprövad metod för verksamhetsutveckling där ledningssystemet samtliga delar; metod, personal, organisation och teknik, utvecklas i harmoni. I beskrivningen av VUM-LS ingår ett stort antal arbetsmetoder, liknande de som beskriv i detta kapitel, standarder, dokemnetation etc som gäller FM ledningssystem utveckling.

Metoden omfatta arbetsflödena:

- Verksamhetsbeskrivning
- Systemspecificering
- Systemdesign
- Systemintegrering
- Systemverifiering

- Systemvalidering



Figur 6. Beskrivning av VUM-LS arbetsflöden: verksamhetsbeskrivning, systemspecificering, systemdesign, systemintegration, systemverifiering och systemvalidering. Under arbetsflödet systemdesign specificeras ledningssystemets övergripande design samt genomförs under samordning parallellt utvecklingsflöden för ledningssystemets delar teknik, organisation, personal och metoder.

Referenser

Akao, Y. (1997). QFD – Past Present and Future. In A. Gustafsson, B. Bergman, & F. Ekdahl (eds.), *Proceedings of the Third Annual International QFD Symposium*, pp. 19-29, vol. 1.

Alexander, I. & Maiden, N. (eds.) (2004) *Scenarios, Stories, Use Cases Through the Systems Development Life-Cycle*. John Wiley.

ANSI/IEEE Std 1471 (2007), *Recommended Practice for Architectural Description of Software-intensive Systems*. http://www.iso-architecture.org/ieee-1471/index.html

Arthur, J. L. (1992). *Improving Software Quality: An Insider's Guide to TQM*. New York: Wiley.

Avison, D. E. & Fitzgerald, G. (1995) *Information Systems Development: Methodologies, Techniques and Tools*. The McGraw-Hill Companies.

Benatallah, B. Sheng, Q. Z., & Dumas, M. (2003) The Self-Serv environment for Web services composition**.** *Internet Computing, IEEE*, 7(1), 40 – 48.

Bergman, B. & Klefsjö, B. (1994) *Quality from Customer Needs to Customer Satisfaction*. London: McGraw-Hill.

Bernard, R. H. (2000) *Social Research Methods. Qualitative and Quantitative Approaches, Sage Publications*. Thousand Oaks, CA.

Boehm, B. W. and Papaccio, P. N. (1998) Understanding and controlling software costs, *IEEE Transactions on Software Engineering*, 14(10), 1462 –1477.

Brooks, F. P. Jr. (1995) *The Mythical Man-month: Essays on Software Engineering*. Addison-Wesley.

Carmel, E., George, J. F., & Nunamaker, J. Jr. F. (1992) Supporting Joint Application Development (JAD) and Electronic Meeting Systems: Moving the CASE Concept Into New Areas of Software

Development. In J. F. Nunamaker (Ed.), *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, 331-342.

Carroll, J. M. (2000) *Making Use. Scenario-based Design of Human-Computer Interactions*, Massachusetts Institute of Technology, MA.

Chaos, The Standish Group, 1995, http://www.standishgroup.com/.

Checkland, P.B. and J. Scholes (2001) Soft Systems Methodology in Action. In J. Rosenhead and J. Mingers (eds), *Rational Analysis for a Problematic World Revisited*. Chichester: Wiley.

Collin B. (2003) *IT-kvalitet: Verksamhets- & Effektivitetsutveckling*. Studentlitteratur, Lund Sweden.

Davenport, T. (1993) *Process Innovation–Reengineering Work through Information Technology*. Boston, MA: Harvard Business School Press.

Davis; A. M. (2005) *Just Enough Requirements Management: Where Software Developments Meets marketing*. Dorset House, New York.

Devanbu, P. T. & Stubblebine, S. (2000) Software Engineering for Security: a Roadmap: The future of Software Engineering, *ACM Press*, 227-239.

Ehn P., Davies, R. C., Brattgård, B., Hägerfors, A., Nilson, J., Dalholm, E. & Mitchell, B. (1996) The Envisionment Workshop - from Visions to Practice. In Blomberg, J., Kensing, F. & Dykstra-Erickson, E. (Eds.) *Proceedings of the Participatory Design Conference*, CPSR, Pao Alto, CA, 141-152.

Flanagan, J. C. (1954) The Critical Incident Technique. *Psychological Bulletin*, 51, 327-58.

Fransson, J. (1999) *Scenariobaserad prototyping av användargränssnitt*, Uppdragsplan MMI- och Funktionalitetsutvärdering, Version 1, FOI.

Friedenthal, S., Moore, A., & Steiner, R. (2008) *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann. Amsterdam.

Gulliksen, J. & Göransson, B. (2002) *Användarcentrerad systemdesign*, Studentlitteratur, Lund.

Hallberg, N, Pilemalm, S., & Westerdahl, L. (2008) *Behovsanalys avseende Försvarsmaktens utveckling av ledningssystem*. FOI 2008, FOI Memo 2443.

Hallberg, N., (1999) *Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach.* [Dissertation No. 596] Linköping Studies in Science and Technology.

Hallberg, N. & Fransson, J. (2001) *UML baserad metamodell för modellering av bekämpningskedjor*. Linköping, FOI 2001, (FOI-R--0159--SE).

Hallberg, N. & Hallberg, J. (2006) The Usage-Centric Security Requirements Engineering (USeR) Method. *Proceedings of the 7th IEEE Workshop on Information Assurance*, U.S. Military Academy, West Point, NY, 21-23.

Hallberg, N., Andersson, R., & Westerdahl, L. (2005) *Quality-driven process for requirements elicitation: the case of architecture driving requirements*. Linköping, FOI 2005, (FOI-R--1576--SE).

Hallberg, N., Ölvander, C., & Törne, A. (2006) *Behovs- och kravanalys avseende tekniskt beslutsstöd för operationer i urban terräng.* Linköping, FOI 2006, (FOI-R--2037--SE).

Hallberg, N., Ölvander, C., Johansson, M., & Törne, A. (2006) *Utvärdering av användningsfall avseende tekniskt ledningssystem för urban miljö*. Linköping, FOI 2006, FOI Memo 1914.

Hallberg, N., Pilemalm, S., & Timpka, T. (1998) Participatory Design of Inter-organizational Systems: A Method Approach. In: *Proceedings of the fifth biennial Participatory Design Conference*, eds. R. Henderson Chatfield, S. Kuhn, and M. Muller. pp. 129-136. (Seattle, WA, 1998)

Hasu, M. & Engeström, Y. (2000) Measurement in Action: an Activity-theoretical Perspective on Producer-user Interaction. *International Journal of Human-Computer Studies*, 53, 61-89.

Hauser, J. R., & Clausing, D. (1988). The House of Quality. *Harvard Business Review*, 63-73.

Hay, D. C. (2003) *Requirements analysis: From business views to architecture*. Prentice Hall, Upper Saddle River, New Jersey

Hull, E., Jackson, K., & Dick, J. (2005) *Requirements Engineering*, 2nd edition, Springer, New York.

IEEE Architecture Working Group, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std 1471-2000, IEEE, 2000.

Jungk, R. & Müllert, N. (1987) *Future workshops: How to Create Desirable Futures*. London, England, Institute for Social Inventions.

Jürjens, J. (2005) *Secure Systems Development with UML*. Springer-Verlag Berlin Heidelberg 2005

Kano, N. (1995) Upsizing the Organization by Attractive Quality Creation. In G. K. Kanji (ed.) *Proceedings of the First World Congress on Total Quality Management*, 60-72.

Karlsson, J. (1998) *Framgångsrik kravhantering: vid utveckling av programvarusystem*. 2(3). Focal point: Linköping.

Kasser, J. E. (2007) *A Framework for Understaning of Systems Engineering*. The right Requirement. Cranflied UK.

Kensing, F., & Munk-Madsen, A. (1993) PD: Structure in the Toolbox. *Communications of the ACM*, 36, 78-85.

Kruchten, P. H. (2004) *The Rational Unified Process. An Introduction*, Addison-Wesley.

Kulak, D. & Guiney, E. (2000). *Use Cases: Requirements in Context*. Addison-Wesley 2000.

Larman, C. (2003) *Agile & Iterative Development*. Addison-Wesley, Boston.

Martin, J. (1991) *Rapid Application Development*. Macmillan.

Mazur, G. (1992) Voice of the Customer Table: A Tutorial. In *Transactions from the Fourth Symposium on Quality Function Deployment*, 105-111.

McGovern, J., Ambler, S. W., Stevens, M. E., Linn, J., Jo, E. K. & Sharan, V. (2003) *The Practical Guide to Enterprise Architecture*. Prentice Hall PTR, New Jersey.

Mizuno, S. (Ed.). (1988) *Management for Quality Improvement: The Seven New QC Tools*. Portland, OR: Productivity Press.

Mizuno, S. & Akao, Y. (eds.). (1994) *QFD: The Customer-Driven Approach to Quality Planning and Development*. Tokyo: Asian Productivity Organization.

MODAF, (2008). Ministry of Defence Architectural Framework http://www.modaf.org.uk/m3/.

Muller, J. K. (1995) Integrating architectural design into the development process. In *Proceedings of the 1995 International Symposium and Workshop on Systems Engineering of Computer Based Systems*, pp. 114 -121.

Muller, M. (1993). PICTIVE: Democratizing the Dynamics of the Design Session. In D. Schuler & A. Namioka (eds*.) Participatory Design. Principles and Practices* (pp. 211-237). Hillsdale, NJ: Lawrence Earlbaum.

Nielsen, J. (1993) *Usability Engineering*, Academic Press, Boston, MA.

OMG (Object Management Group), *Unified Modeling Language Specification*, Version 1.4, 2001.

Pilemalm, S., Lindell, P.O., Hallberg, N., & Eriksson, H. (2007) Integrating the Rational Unified Process and participatory design for development of socio-technical systems: a user participative approach. *Design Studies*, 28(3), (May 2007), pp 263-288.

Rabhi, F. & Benatalla, B. (2002) An integrated service architecture for managing capital market systems. *IEEE Networks*, vol 1.

Schekkerman, J. (2003) *How to survive in the jungle of enterprise architecture framework: Creating or choosing an enterprise architecture framework*, Trafford, Victoria, BC, Canada.

Schuler, D. & Namioka, A. (eds.) (1993). *Participatory Design: Principles and Practices*. Hillsdale, NJ: Lawrence Earlbaum.

Siddiqi, J. & Shekaran, M.C. (1996) Requirements Engineering: The Emerging Wisdom, *IEEE Software*, 13, 1996, pp. 15-18.

Sindre, G. & Opdahl, A. L. (2000) Eliciting Security Requirements by Misuse Cases, TOOLS Pacific 2000: 37th International Conference on Technology of Object-Oriented Languages and Systems, Sydney, Australien.

Sommerville, I. & Sawyer, P. (1997). *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons

Sommerville, I. (2001) *Software engineering*. Addison-Wesley, 2001

Sommerville, I., Sawyer, P. & Viller, S. (1998) Viewpoints for requirements elicitation : a practical approach. *In Proceedings of the Third IEEE International Conference on Requirements Engineering*, Colorado Springs, Colorado, USA.

Sutcliff, A. (2002) *User-Centered Requirements Engineering. Theory and Practice*. Springer, London

The open group, (2006). The open group architectural framework (TOGAF). http://www.theopengroup.org/architecture/togaf8-doc/arch/

Wiegers, K. E. (2003) *Software requirements*. Microsoft press, Redmond WA.

Wood, J. & Silver, D. (1995) *Joint Appplication Development*. New York, Johan Wily and Sons.

Young, R. (2001) *Effective Requirements Engineering*. Addison-Wesley.

Zachman, J. A. (1987) A Framework for Information Systems Architecture, *IBM Systems Journal*, 26(3), 276-292.