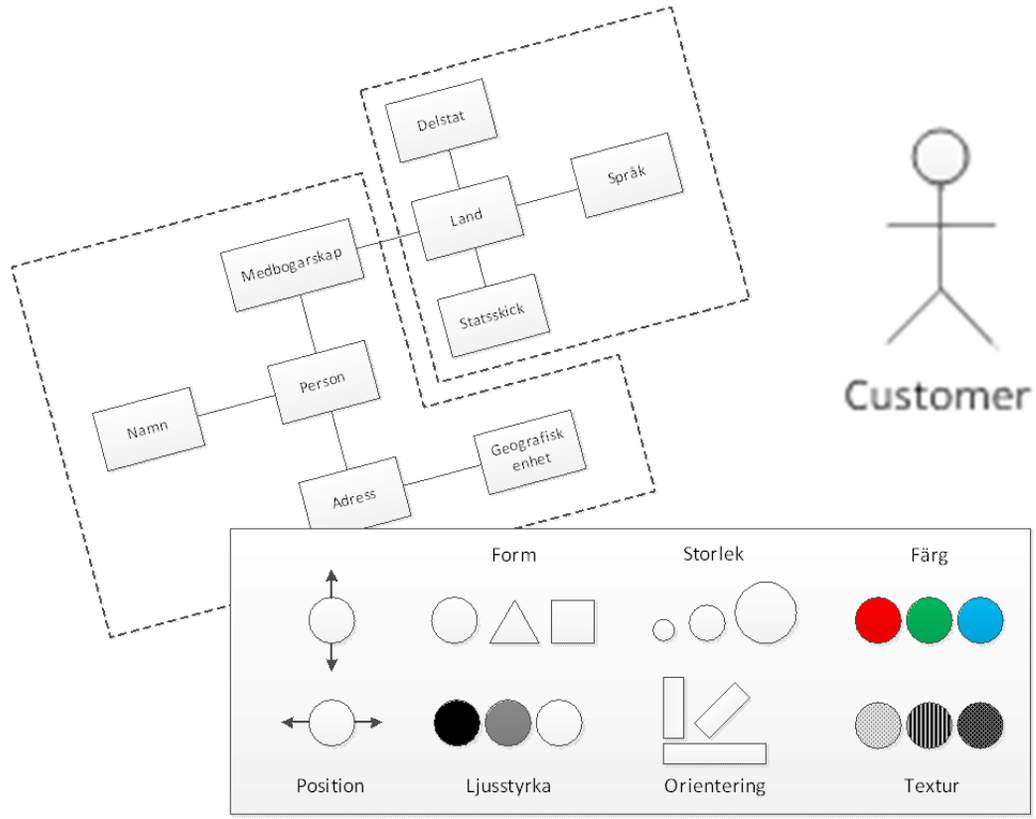
Kvalitetsbaserad ledningssystemutveckling: Metoder och principer

NIKLAS HALLBERG, NINA LEWAU, JOACHIM HANSSON,
HELENA GRANLUND, SUSANNA NILSSON,
JONAS HARALDSSON, HENRIK KARLZÉN



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--3358--SE
ISSN 1650-1942

Användarrapport
December 2011

Informationssystem

Niklas Hallberg, Nina Lewau, Joachim Hansson, Helena Granlund, Susanna Nilsson, Jonas Haraldsson, Henrik Karlzén

Kvalitetsbaserad ledningssystemutveckling: Metoder och principer

Titel	Kvalitetsbaserad ledningssystemutveckling: Metoder och principer
Title	Quality-based Development of C^2 Systems: Methods and Principles
Rapportnr/Report no	FOI-R--3358--SE
Rapporttyp/Report Type	Användarrapport/User report
Månad/Month	December/December
Utgivningsår/Year	2011
Antal sidor/Pages	89 p
ISSN	ISSN 1650-1942
Kund/Customer	Försvarsmakten
Projektnr/Project no	E53334
Godkänd av/Approved by	Magnus Jändel

FOI, Totalförsvarets Forskningsinstitut	FOI, Swedish Defence Research Agency
Avdelningen för Informationssystem	Information Systems
Box 1165	Box 1165
581 11 Linköping	SE-581 11 Linköping

Sammanfattning

Kvalitetsbaserad systemutveckling handlar om att utveckla system som håller hög kvalité och dessutom att göra detta så kostnadseffektivt som möjligt. Detta är ett mål som eftersträvas i all form av systemutveckling. För att lyckas med kvalitetsbaserad utveckling och nå detta mål, finns ett flertal ansatser inom olika områden. I denna rapport beskrivs sju ansatser för att erhålla kvalité i utvecklingen av system: (1) Lean, (2) Software Product Line Engineering (SPLE) (3) Six Sigma, (4) Capability Maturity Model Integration (CMMI), (5) Goal Oriented Requirements Engineering (GORE), (6) Kvalitetssäkring av krav och (7)The physics of notations.

Samtliga dessa ansatser ämnar till att på olika sätt främja kvalitén inom systemutveckling. Målet för Lean är att göra utvecklingen så effektiv som möjligt bland annat genom att reducera aktiviteter som inte bidra till kvalité hos system. SPLE har utvecklats med målsättningen att förbättra och effektivisera utvecklingen av mjukvara genom att återanvända istället för att nyutveckla för varje konfiguration och variant av system, till exempel genom att skapa programsviter där samma komponent nyttjas av flera olika program. GORE är en bred ansats som har fokus på målorienterad kravhantering och det finns en mängd olika metoder och notationer inom ramen för ansatsen.

Utöver dessa ansatser beskriver rapporten även ansatser som har mer övergripande fokus på kvalité under hela utvecklingsprocessen. Six Sigma är inriktat på att följa upp och mäta kvalité i utveckling, medan CMMI är ett standardiserat sätt att mäta hur väl organisationer lyckas med utveckling. Ett sätt att studera hur väl en utvecklingsprocess fungerar ur effektivitets- och kvalitetsperspektiv är att granska och kvalitetssäkra de krav som är ingångsvärden i systemutvecklingen och även denna ansats beskrivs och diskuteras. En viktig del i systemutveckling är dessutom notationer, eller de språk som används för att modellera och beskriva systemet och processen. I ”the physics of notations” diskuteras principer för hur notationer bör ska utformas för att vara kognitivt effektiva.

Samtliga dessa ansatser är av intresse för Försvarsmakten att beakta, och efter anpassning anamma delar av eller i helhet. Vilka ansatser och i vilken form som skulle fungera i Försvarsmakten utgör dock grunden för vidare arbete.

Nyckelord: Systemutveckling, kvalitetsbaserad, processutveckling, kravhantering, modellering

Summary

Quality-based systems development is about developing systems that are of high quality and to do so as cost effectively as possible. This is a goal strived for in all kinds of systems developments. To succeed with quality-based development and achieve this goal several approaches are used in different areas of development. This report describes seven approaches to achieve quality in the development of systems: (1) Lean, (2) Software Product Line Engineering (SPLE), (3) Six Sigma, (4) Capability Maturity Model Integration (CMMI), (5) Goal Oriented Requirements Engineering (GORE), (6) quality assurance of requirements, and (7) the physics of notation.

All of these approaches intend to promote quality in systems development. The goal of Lean is to make development as efficient as possible, by reducing activities that do not contribute to the quality of the to-be system. SPLE has been developed with the aim to improve and streamline the development of software by creating a suite of programs where the same components are used by several applications. Instead of develop new items the developers are able to bootstrap new configurations of the system by reuse. GORE is a broad approach that focuses on goal-oriented systems development and there are a variety of methods and notations in the context of GORE.

In addition to these approaches, the report also describes approaches that do not have as its main focus to shape the actual development process, but that has a comprehensive focus on quality. Of these, Six Sigma focuses on monitoring and measuring the quality of development, while CMMI is a standard way to measure how well organizations succeed in development. This report also describes a method to study the development process in terms of efficiency and quality by reviewing and quality assure the requirements that are the initial values of system development. An important topic in systems development is the notations used to model and describe the systems and processes. Physical principles for how modelling notations can be visualized more cognitively effective are discussed in the report.

These approaches are all potentially of interest to the Swedish Armed Forces, and after adjustment possible to embrace in part or in whole. What approaches and in what form that would serve the Swedish Armed Forces are, however, the basis for further work.

Keywords: Systems development, quality based, process development, requirements engineering, modeling

Innehållsförteckning

1 Inledning

Att utveckla system så att dessa håller *hög kvalité* är eftersträvansvärt, men har visat sig vara en icke-trivial uppgift (Kasser, 2007). De svårigheter och utmaningar som finns i systemutveckling, existerar oavsett vilken typ av system det rör sig om, exempelvis informationssystem, materielsystem, personalförsörjningssystem och verksamhetssystem. 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, ses ledningssystem som bestående av människor och tekniska stöd, vilka är organiserade och följer givna arbetsprocedurer. När det gäller svårigheter och utmaningar vid utveckling är ledningssystem inget undantag, utan är snarare ännu mer komplicerade att utveckla då dessa är relativt komplexa, innefattande människor, organisationer, verksamhetsprocesser och teknik. Kvalitén i utvecklade system beror i stor utsträckning på de ansatser som tillämpas vid utvecklingen. En utmaning inom systemutveckling är dock att det inte finns någon enhetligt *bästa* process, utan hur denna process ser ut är beroende på förutsättningarna samt vilken typ av system som ska utvecklas. Syftet med denna rapport är att beskriva ett antal olika ansatser (metoder och principer) för att erhålla kvalité vid utveckling av system. De ansatser som beskrivs är:

- Lean
- Software Product Line Engineering (SPLE)
- Six Sigma
- Capability Maturity Model Integration (CMMI)
- Goal Oriented Requirements Engineering (GORE)
- Kvalitetssäkring av krav
- The physics of notations

Urvalet av ansatserna som studerats i denna rapport ska ses som en fortsättning på det arbetet som genomförts inom FoT-projektet Arkitekturbaserade ledningssystemutveckling. Där *Lean* och *Six Sigma* har kopplingar till Quality Function Deployment och Kvalitetsdriven kravhantering som beskrivs i Hallberg, Pilemalm, Westerdahl et al (2008). *GORE* och *Kvalitetssäkring av krav* är ett sätt att ytterligare förbättra kravhanteringen som lyfts fram i den behovsanalys avseende utveckling av ledningssystem i Hallberg, Pilemalm, Westerdahl et al (2008). Kvalitetssäkring av krav utvecklades också ursprungligen i FoT-projektet Arkitekturbaserade ledningssystemutveckling. *SPLE* och *The physics of notations* ska ses som vidare utveckling av

modellbaserad utveckling (Hallberg, Pilemalm, Sparf et al, 2009). *Lean*, *Six Sigma* och *CMMI* strävar samtliga till att förbättra utvecklingsprocessen.

En av de absolut mest kritiska aktiviteterna att lyckas med i systemutveckling är att identifiera, beskriva och kommunicera de krav som ska ligga tillgrund för utformningen av system (Boehm & Papaccio, 1998; Wiegers, 2003; Young, 2001). Dessa ansatser syftar främst till att bygga rätt system. Nedanstående ansatser som beskrivs i rapporten berör detta.

- GORE är en ansats som fokuserar på målen för systemet, för att sedan koppla målen till kraven.
- Kvalitetssäkring av krav är en metod för att ge stöd vid formulering och granskning av krav.

För att under systemutveckling utnyttja tillgängliga resurser så effektivt som möjligt och leverera system inom givna ramar krävs att processerna ständigt förbättras (Bergman & Klefsjö, 1994; Ficalora & Cohen, 2010). Detta kan göras genom att mäta, utvärdera, identifiera och eliminera defekter samt att skapa en förståelse och en kultur för att fokusera på kvalité. Kvalité kan även åstadkommas genom att systematiskt återanvända redan befintliga komponenter. Nedanstående ansatser syftar till att utveckla system på rätt sätt:

- CMMI är ett ramverk för att skapa mätbara, förutsägbara och kvalitetssäkrade processer.
- Six Sigma är problemorienterat och syftar till att eliminera problem i processen.
- Lean bygger på ett antal principer främst för att eliminera slöseri i processer.
- SPLE är en ansats för att hantera en gemensam bas av funktioner och på den ett antal produkter med egna funktioner.
- The physics of notations består av nio principer för att skapa kognitivt effektiva visuella notationer, det vill säga representationen av modeller.

Denna rapport är den andra från det FoT-finansierade projektet Kvalitetsbaser*ad ledningssystemsutveckling*. Målet med projektet är att utveckla vetenskapligt kvalitetssäkrad och praktiskt förankrad kunskap, samt utveckla kompetens avseende kvalitetsbaserad ledningssystemsutveckling. Utgångspunkten för projektet är att nyttja utvecklingsansatser som bygger på vedertagna principer för att uppnå kvalité vid utveckling.

2 Bakgrund

Detta avsnitt beskriver (1) begreppen kvalité och kvalitetssäkring, (2) kravhantering och (3) visuella notationer.

2.1 Kvalité och kvalitetssäkring

Kvalité är ett mångtydigt begrepp. Enligt svenska akademins ordlista[1] har begreppet *kvalité* fem olika betydelser; (a) värde, (b) egenskap, (c) sort, (d) beskaffenhet och (e) god beskaffenhet. I engelska Merriam-Webster ges begreppet *quality* (sve. kvalité) åtta olika betydelser[2].

Även inom systemutvecklingsområdet används begreppet kvalité (eng. quality) med olika betydelser. Inom kravhanteringsområdet förekommer begreppet *quality requirements*, vilket motsvarar icke-funktionella krav (Nuseibeh & Easterbrook, 2000). I detta sammanhang är *quality* liktydigt med egenskaper. Det vill säga att *quality requirements* kravställer egenskaper hos system. I den japanska traditionen av kvalitetstänkande med ursprung från sextiotalet handlade kvalité om att säkerställa *värdet* för kunder (Mizuno & Akao, 1994). Denna syn spreds snabbt relativt snabbt och *kvalité* kom att innebära:

- inte göra något fel betyder inte att göra något rätt (Braithwaite, 1993)
- undvika defekter är inte längre tillräckligt (Deming, 1994)
- produkter ska passa sitt syfte (Juran, 1988)
- produkter ska möta konsumenternas nuvarande och framtida behov (Deming, 1986)
- produkters kvalité är deras förmåga att tillfredställa kunders behov och förväntningar (Bergman & Klefsjö, 1994)

Six Sigma, Total Quality Management (TQM) och Quality Function Deployment (QFD) är exempel på ansatser som baseras på detta tänkande (Ficalora & Cohen, 2010; Bergman & Klefsjö, 1994; Akoa, 1997). Målet med TQM är att skapa en organisation (inkl. ledning) som främjar att producera värde för kunder. QFD är

[1] http://www.svenskaakademien.se/svenska_spraket/svenska_akademiens_ordlista/saol_pa_natet/ordlista

[2] Quality enligt Merriam-Webster: (1) peculiar and essential character, (2) degree of excellence, (3) social status, (4) distinguishing attribute, (5) the character in a logical proposition of being affirmative or negative, (6) vividness of hue, (7) timbre and the identifying character of a vowel sound determined chiefly by the resonance of the vocal chambers in uttering it and (8) the attribute of an elementary sensation that makes it fundamentally unlike any other sensation (www.m-w.com, 2011-11-16)

en ansats för att säkerställa att utvecklade produkter, tjänster etc. motsvarar kundernas behov. Detta åstadkoms genom att säkerställa att funktioner och egenskaper hos det som utvecklas bidrar till att möta kunders behov. I och med detta kom kvalité inte längre att enbart vara en *objektiv* utan även, och kanske främst, en *subjektiv* företeelse. En objektiv kvalité finns hos systemet medan den subjektiva kvalitén är upplevelsen av hur väl det fungerar. Att avgöra om något har hög kvalité kan vara svårare att påvisa än bristande kvalité. *Kvalité* i systemutveckling handlar inte enbart om att bygga rätt system, det ska också byggas på rätt sätt och rätt första gången (Arthur, 1992). Med kvalitetsbaserad avses i detta projekt att:

1. **Utveckla rätt system**. Det vill säga system som åstadkommer värde för dess intressenter. De ska enbart innehålla funktioner och egenskaper som bidrar till värdet för intressenterna.
2. **Utveckla system rätt**. Det vill säga att genomföra systemutvecklingen på ett kostnads- och resurseffektivt sätt. Detta genom att bygga rätt system på rätt sätt första gången.

2.2 Kravhantering

Kravhantering kan ses som en mekanism för att säkerställa att användares och verksamheters behov tillgodoses av de system som utvecklas. Att lyckas med kravhantering är dock långt ifrån trivialt. Krav måste identifieras, analyseras, valideras och dokumenteras samt kommuniceras till dem som ska utforma systemet. Konrad och Gall (2008) hävdar att det största skälet till att utvecklingsprojekt misslyckas är otillfredsställande kravhantering. Vidare påstår Young (2001) att den största andelen av de felaktigheter som finns i befintliga system kan härledas till brister i kravhanteringen. En av dessa är avsaknaden av kunskap och kompetens att utforma korrekta kravspecifikationer, många kravspecifikationer innehåller en rad felaktigheter. Enligt Firesmith (2003) är de vanligaste felen att kravspecifikationer är *tvetydiga*, *ofullständiga*, *inkonsistenta*, *felaktiga*, *ogenomförbara* samt innehåller krav som inte går att *verifiera*. Ett sätt att undvika sådana undermåliga kravspecifikationer är att utsätta dem för en rigorös granskning. Kravgranskning kan genomföras med två olika syften, kvalitetssäkring av innehåll respektive form (Hansson, Granlund, Hallberg, et al., 2010). En innehållsgranskning genomförs för att säkerställa att krav är korrekta, att de motsvarar de behov som avses tillgodoses med systemet. Granskning utifrån form beaktar hur krav är formulerade, det vill säga att formuleringarna följer givna kriterier. Detta senare kan upplevas som mindre viktigt, men korrekta krav som är felaktigt formulerade kan likväl leda till felaktiga designbeslut. Att granska krav är en tidskrävande uppgift men ger vinster i form av tydligare kravspecifikationer som in sin tur skapar förutsättning för att slutliga system innehåller färre felaktigheter.

2.3 Visuella notationer

Det finns ett talesätt som säger att ”en bild säger mer än tusen ord”. Men det finns också de som menar att ”tusen ord säger något annat än en bild”[3]. Oavsett vilket av påståendena som gäller så används en fjärdedel av vår hjärna för att processa visuell information, vilket är mer än de andra sinnena tillsammans. Följaktligen är visuell information ett effektivt sätt för oss människor att ta till sig information.

Inom informationssystemutvecklingen har visuella notationer, i form av modeller och diagram, använts sedan systemutvecklingens begynnelse och utgör ett viktigt medel i utvecklingen. Den första visuella notationen inom utvecklingen av informationssystem var Goldstine och von Neumanns flödesbild från 1940 (Goldstine & von Neumann, 1947/1963). Trots människans effektivitet när det gäller att ta till sig visuell information har den visuella delen av notationer fått ytterst lite uppmärksamhet i forskning och tillämpning, och har således blivit ett eftersatt område. Den största delen har ägnats åt den semantiska delen av notationer, det vill säga VAD en notation ska kunna uttrycka (roll, aktivitet, mål, förmåga etc.), och inte HUR detta ska göras. Att lyckas förmedla sitt budskap med en bild, ett diagram eller modell är ingen lätt utmaning. Det finns forskning som påvisar att den visuella delen av notationen är (minst) lika viktig som den semantiska för förståelsen av information i diagram (Moody, 2009).

Frågor kring hur människor uppfattar och tolkar bilder och andra visuella representationer har studerats inom flera domäner, bland andra psykologi, neurovetenskap och olika grenar av formgivningsområdet såsom media och grafisk design. Visuellt språk kan användas för flera syften, till exempel för att styra en mottagare av ett budskap eller en användare av en hemsida i en viss riktning, att förtydliga eller förstärka ett verbalt budskap eller som estetiskt tillägg. Forskning inom användbarhetsområdet har bland annat lett till utvecklandet av riktlinjer för design och utformning av exempelvis webbplatser och användargränssnitt. Nielsen (1993) utvecklade 10 heuristiker för användbarhet, Shneiderman (1998) utformade ”8 golden rules of interface design” och Norman (1998) beskrev generella designprinciper som rör både fysiska och digitala artefakter. Alla dessa riktlinjer och principer poängterar vikten av att ta hänsyn till, och utnyttja, människans kognitiva förmågor.

Inom systemutvecklingsområdet har forskningen inom den visuella delen av notationer varit relativt eftersatt men det görs forskning inom området. Moody (2009) har gjort ett försök att skapa en grund för att utvärdera, jämföra och skapa

[3] Citat av journalisten Göran Rosenberg som i sin helhet lyder: ”Lika lite som en bild säger mer än tusen ord, säger tusen ord mer än en bild. Tusen äpplen smakar inte mer än ett päron och tusen päron smakar inte mer än ett äpple. Äpplen smakar något annat än päron hur många de än är. Tusen ord säger något annat än en bild” (Bergström, 2004, s 261).

effektiva visuella notationer. Idag tas många notationer fram utifrån intuition och personlig smak, ej utifrån vad som är kognitivt effektivt. Moody kallar sin teori för ”the physics of notations”, eftersom den rör den fysiska delen av notationen, det vill säga hur den ser ut, och inte den logiska delen, det vill säga vad den ska representera. Moodys hävdar att hans syfte inte är att underminera existerande notationer eller föreslå nya, utan att bidra till att öka (den kognitiva) effekten av modellerna och nyttja den kunskap som finns om hur människor tar emot och bearbetar information (kognition). Enligt Moody (2009) är de främsta fördelarna med att göra modeller mera kognitivt effektiva att:

- målgrupper på ett effektivt sätt tar till sig och förstår informationen som förmedlas i modellen.
- modellen ”håller ihop”, det vill säga att notationen och utformningen är konsekvent så att läsaren inte behöver ”lära om” för varje diagram i modellen.
- kognitivt effektiva modeller medför att det är lättare bedöma om syftet med modellen har uppfyllts.

Utöver dessa punkter gör kognitivt effektiva modeller det lättare för en ny användare/medarbetare att sätta sig in i modellerna och det formspråk som används i notationen.

3 Kvalitetsbaserade ansatser

I detta kapitel beskrivs sju ansatser till metoder och principer för att erhålla kvalité vid utveckling av system (Tabell 1). Det vill säga dessa handlar så väl om att *utveckla rätt system* som att *utveckla system rätt*.

Tabell 1 De ansatser som beskrivs i rapporten

Ansatser	Beskrivning
Lean	Förhållningssätt för att fokusera på att skapa kundvärde och anpassa sina processer för att minimera resursslöseri.
Software Product Line Engineering (SPLE)	Utvecklingsmetodik för mjukvara där långtgående återanvändning av befintliga komponenter nyttjas. Programsviter skapas där flera olika konfigurationer av programmet med olika ändamål baseras på samma grundfunktionalitet.
Six Sigma	Ansats för att minska antalet misstag som görs inom organisationen. Six sigma används i olika grader från en enskild metrik till ett heltäckande förhållningssätt för hela organisationer.
Capability Maturity Model Integration (CMMI)	Samling metoder för att bedöma hur väl utvecklingsprocessor fungerar inom en organisation. Organisationer graders enligt mognadsgrad och svagheter identifieras som behöver hanteras för att nå en ökad mognad.
Goal-oriented requirements engineering (GORE)	Ansats inom kravhantering där fokus inte är på *hur* ett system ska fungera utan *varför* och tillsammans med *vilka*.
Kvalitetssäkring av krav[4]	Metod för att granska och kvalitetssäkra formuleringen av enskilda krav. Granskning sker på en rent språklig nivå och värderar krav utifrån åtta attribut.
The physics of notations	Nio principer som syftar till att öka den kognitiva effektiviteten hos visuella modeller.

[4] I denna rapport beskrivs resultat av två fallstudier där metoden tillämpades.

3.1 Lean

Under slutet av 1950-talet erfor USA:s bilindustri, Ford, GM och Chrysler, hur japanska biltillverkare, speciellt Toyota, lyckades ta plats på den amerikanska marknaden. Ford hade 40 år tidigare revolutionerat industrin genom att bland annat utveckla ett arbetssätt baserat på löpande band, den så kallade Taylorismen. I arbetssättet ingick att anställda arbetade inom ett begränsat område med enkla begränsade uppgifter. Produkten som framställdes rörde sig genom lokalen och förbi de anställda. Kunskap arbetades in i processen i stället för att krävas av de anställda. Der kvalificerade arbetet flyttades från hantverkaren till designern. Uppdelningen mellan professionerna *industriarbetare* och *ingenjör* ökade. Då kravet på utbildning minskade ökade mängden tillgänglig arbetskraft, däribland de amerikanska kvinnorna som varit relativt skyddade från krigets verkningar, och stora kvantiteter av avancerade produkter kunde tillverkas. Löpande bandet var en utveckling som inte bara påverkade utformningen av industriprocesser utan hela det västerländska samhället. USA hade möjlighet att under världskrigen utveckla processer. Europa, som allvarligt påverkades av krigen, anammade USA:s process-kunskap utan större egna reflektioner och anpassningar.

Japans förutsättningar var annorlunda. Framförallt fanns inget etablerat samarbete med USA och efter andra världskrigets dramatiska slut var det heller inte aktuellt. Japansk industri gick sin egen väg. Toyota började efter andra världskrigets slut utveckla en industriprocess grundläggande skild från löpande bandet. I Toyotas process arbetade de anställda samlat runt produkten som var stilla i jämförelse med löpande bandets produkter. De anställda gavs möjlighet att bilda sig en uppfattning om meningen med produkten och möjlighet att själva förbättra processen. Jämförelsevis små kvantiteter producerades, men med högre kvalitet, där arbetssättet uppmuntrade till utbildning och motivation. Processen hade kortare ledtider, högre produktflexibilitet, bättre kundförståelse, högre produktivitet och utnyttjande av utrustning än det löpande bandet (Liker, 2004). Toyotas anställda utvecklade en medvetenhet för produktens kvalitet, medan amerikanska biltillverkares anställda förblev medvetna endast om sina egna begränsade sfärer. Företaget Toyota såg de anställda som en utbildad tillgång, medan amerikanska biltillverkare såg den stora massan av anställda som utbytbar. Toyotas produkt vann i kvalitet på ett märkbart sätt för kunden och marknadsandelar började vinnas i väst. De grundläggande skillnaderna i ansats för de två processerna utmynnade i en explosion av forskning och utveckling av industriprocesser generellt.

De grundläggande begreppen i Toyotas ansats samlas tillsammans med liknande förhållningssätt under termen Lean (eng. Lean Production och Lean thinking). Huvuddraget i Lean är att se till hela organisationen, dels identifiera vilket kundvärde den ska producera och dels identifiera vilka slöserier den utför. När

slöserierna är identifierade ska de elimineras på ett sådant sätt att det totala kundvärdet ökar. Stommen i de metoder som används för att eliminera slöserier sammanfattas som ofta som Lean-principer. Utveckling av Lean har pågått sedan 1960-talet. Förhållningssättet sprider sig stabilt till domäner andra än processindustri. Det här avsnittet avser att beskriva Lean och klassiska Lean-principer parallellt med Leans möjligheter i området systemutveckling.

3.1.1 Lean systemutveckling

Lean har spridits via processindustrin till en mängd andra områden. För varje ny domän behöver grunderna för förhållningssättet, kundvärde och slöseri, identifieras specifikt för den nya domänen. Exempel på områden förutom processindustri som arbetar med Lean är sjukvård (Kim, Spahlinger, Kin & Billi, 2006), offentlig sektor (Radnor & Walley, 2008), mjukvaruutveckling (Poppendieck & Poppendieck, 2003) och systemutveckling (Oppenheim, 2011).

Systemutveckling är en domän med en stark ingenjörsmässig identitet och teknisk kärna. För systemutveckling gäller att "inget faller mellan stolarna" avseende teknisk prestanda, interna och externa gränssnitt, kostnader och tidplan, operativa behov och regelverk. Detta berör speciellt aktiviteter under systemutvecklingens tidiga faser. Traditionell Lean däremot lägger vikt på kvalitet, inflytande och människors förmåga att utföra processaktiviteter, smidigt informationsflöde, kontinuerliga förbättringar, och maximering av mervärdet av varje aktivitet, det vill säga aktiviteter i de sena faserna av produktutveckling.

Systemutveckling fokuserar på produktutveckling, emedan Lean fokuserar på produktion. Systemutveckling är fokuserad på de aktiviteter som leder till den definition av produkten som är mest sannolikt att framgångsrikt möta kundernas behov. Lean fokuserar på de aktiviteter som leder till förverkligandet av den produkt som med framgång kommer att ge kunden ett värde.

De uppenbara skillnaderna gör att systemutveckling och Lean upplevs svåra att associera med varandra och *Lean systemutveckling* är ett ungt område (Oppenheim, 2011). En sammanslutning som under ett tiotal år varit övertygade om systemutveckling och Leans möjliga kombination är Lean Advancement Initiative (LAI), ett amerikanskt konsortium som samlar deltagare från amerikanska myndigheter såväl som industri och universitet. Konsortiet har en delad tro på att Leans principer och praxis ger en effektiv strategi med målet att skapa värde och minska slöseri speciellt vid systemutveckling (Rebentisch, Rhodes & Murman, 2004; LAI, 2011). I LAI:s nätverk för utbildning, EdNet, myntades 2003 begreppet Lean systemutveckling (eng. Lean system engineering) (Oppenheim, Murman & Secor, 2009).

INCOSE, International Council on Systems Engineering, har erkänt och definierat lean systemutveckling som tillämpningen av Leans principer, metoder och verktyg inom domänen systemutveckling för att öka mängden levererat värde för systemets intressenter (INCOSE, 2009).

3.1.2 Värde

I sin enklaste form består värde av det kunden upplever är viktigt och är villig att betala för. Detta fungerar enkelt vid köp av glass, men är avsevärt svårare vid upphandling av ett nytt komplext tekniskt system (Oppenheim, 2011). Svårigheten i att identifiera värde ökar med antalet intressenter och uppgifter som ska utföras. Värdet förändras dessutom med tiden. För traditionell Lean är kundvärde fundamentalt. Kundvärdets förändring omhändertas i förhållningssättet i och med inställningen att en organisation ständig ska förbättras.

3.1.3 Slöseri

Traditionell Lean strävar efter att alla aktiviteter som utförs i en organisation ska leda till ett kundvärde. Det innebär att organisationen behöver klargöra vad som har ett värde, men även identifiera vad som inte har ett värde. De aktiviteter som försiggår i organisationen och som inte leder till ett värde ses som slöserier. Toyota identifierade sju typer av slöserier som stöd för organisationer med huvudsakligen processindustri-baserad verksamhet (Womack & Jones, 2003):

1. Onödig transport av produktionsdelar.
2. Att personal förflyttar sig i onödan.
3. Extra-arbete så som omarbetning.
4. Onödig lagring av produkter eller produktdelar som inte omedelbart går vidare i flödet.
5. Överproduktion så som att producera mer än nästa steg kan ta emot.
6. Stopp i flödet på grund av väntan på att det tidigare steget ska bli klart.
7. Kvalitativa eller behovsmässiga defekter och misstag.

Flera kompletterande typer av slöseri har föreslagits, exempelvis bortkastad talang (Oppenheim, Murman & Secor, 2009). Från slöserierna vidareutvecklades principer för hur slöserierna kan arbetas bort ur organisationen.

När förhållningssättet Lean införs i ett nytt område behöver berörd organisation ta ställning till vilka slöserier som finns i organisationen. Mycket i Lean är baserat på sunt förnuft och kan återanvändas direkt i andra komplexa domäner än processindustri. Berörd organisation har därmed hjälp i att ta lärdom av de

principer som redan utarbetats (Locher, 2011). Ett område skilt från processindustri som kan tas som exempel på hur identifiering av slöserier kan leda till kvalitetshöjande arbetssätt är Poppendieck och Poppendiecks (2003) arbete inom mjukvaruutveckling. Mjukvaruutveckling ses som ett specialfall av systemutveckling där värde identifieras av analys och kodning. Med dessa två i åtanke som värdeskapande aktiviteter identifierades sju motsvarigheter till Toyotas slöserier specifika för mjukvaruutveckling (Poppendieck & Poppendieck, 2003):

1. Delvis utfört arbete (motsvarande onödig lagring)
2. Extra funktioner (motsvarande överproduktion)
3. Omlärning (motsvarande extra-arbete)
4. Överlämnande (motsvarande att personal förflyttar sig i onödan)
5. Byte av uppgift (motsvarande transport av produktionsdelar)
6. Väntan på grund av förseningar (motsvarande stopp i flödet)
7. Defekter (motsvarande defekter)

Poppendieck och Poppendieck (2003) vidareutvecklade ett tjugotal principer från dessa sju slöserier. Dessa principer kan användas som stöd för organisationer med huvudsaklig verksamhet i mjukvaruutveckling i arbetet med att identifiera och eliminera slöserierna i enlighet med förhållningssättet Lean.

3.1.4 Lean-principer

Lean-principer utgör grunden i de metoder olika organisationer använder för att komma tillrätta med slöserier. Metoderna växer fram som ett resultat av alla anställdas ansträngning för förbättring. Det innebär en bottom-up ansats där de anställda är botten och det slutliga kundvärdet är toppen. Metoderna är inte definierade i förväg, men de kan dokumenteras i efterhand i form av principer. Dokumentering har gjorts av ett flertal författare där principerna är mer eller mindre överlappande beroende på författarens intresse och organisationens domän. Till exempel Locher (2011) beskriver en uppsättning principer för kontorsarbete arbete och Womack och Jones (2003) beskriver fem principer för processindustri. I det här avsnittet beskrivs sex Lean-principer för systemutveckling, samt den klassiska beskrivningen av Toyotas 14 företagsledningsprinciper (Oppenheim, 2011; Liker 2004).

3.1.4.1 Sex principer för systemutveckling

Lean handlar om att leverera det värde som kunder efterfrågar utan att förlora respekt för någon intressent. Genom att klart definiera värdet av en speciell produkt eller tjänst ur slutkundens perspektiv kan det slöseri som inte skapar värde för kunden identifieras och tas bort. I det här avsnittet beskrivs sex

principer för Lean systemutveckling, där varje princip har ett antal *enablers* (motsvarande: Möjliggörare): *Lean Enablers for System Engineering*, vilket har utvecklats av INCOSE:s arbetsgrupp för Lean systemutveckling (Oppenheim, 2011).

Value

Principen Value (sve. värde) syftar till att säkerställa att en heltäckande, tydlig och detaljerad förståelse för vilket värde kunden efterfrågar finnas i hela den systemutvecklande organisationen. Detta för att framgångsrikt utveckla rätt produkt. Value innebär mer än att alla ska ha tillgång till en klassisk kravdokumentation utan kräver även förståelse för kundens verksamhetskultur och behov, liksom förståelse för systemets interoperabilitets- och kompabilitetskrav. Denna första princip kräver en robust process (Oppenheim, 2011). De möjliggörare som krävs är främst att definiera värdet på produkten eller systemet för kunden, samt att frekvent involvera kunden (Oppenheim, 2011).

Map the value stream

Principen Map the value stream (sve. kartlägg värdeflödet) menar att de aktiviteter som skapar värdet noggrant måste planeras. Dålig förberedelse och planering skapar en enorm potential för slöseri. Enligt principen innebär den första delen att få en överblick över samtliga processer och därefter eliminera onödiga processteg och undvika att personal väntar på annan personal (Womack & Jones, 2003).

De möjliggörare som krävs är bland andra (Oppenheim, 2011):

- att skapa många möjliga lösningar och ta sent beslut om specifik lösning och säkerställa ett snabbt genomförande när beslutet om lösning är taget,
- att planera för att lägga mycket tid och resurser i början av projektet för att säkerställa att projektet gör rätt saker när det väl startar,
- att enbart utveckla det som måste utvecklas exempelvis genom att återvinna,
- att samverka med och utbilda sina leverantörer för att undvika potentiella konflikter
- planera för vilka metriker som ska användas för att kunna leda systemets utveckling baserat på kunskap om effekter på kundvärdet.

I den här principen ligger även utformning av lokala planer för att effektivisera arbetet för personal. Individer som är nödvändiga i värdeflöden ska identifieras och ges möjlighet att göra sitt bästa i alla processer, liksom att aktiviteter som verkar negativt på andra värdeströmmar ska identifieras och åtgärdas (Locher, 2011).

Flow

Principen Flow (sve. flöde) handlar om att skapa flöde och ett flyt i processerna, för att undvika att personalen får vänta på information eller överproducerar, vilket i systemutvecklingsprojekt är de vanligaste aktiviteterna (Oppenheim, 2011). Det krävs även här noga planering för att undvika stopp och omarbetningar. Den möjliggörare som krävs för att uppfylla detta är bland annat (Oppenheim, Murman & Secor, 2009):

- Att ofta förtydliga och prioritera krav samt att ofta skapa möjlighet för beslut är vitalt för att kunna balansera flödet.
- Att tidigt lägga mycket tid och resurser på att ta fram arkitektur och design förbättrar beslutsmöjligheter och ökar flexibilitet.
- Att göra överproduktion och flaskhalsar synliga för alla med effektiv kommunikation och enkla verktyg förbättrar personals möjlighet att arbeta med flöde.
- Viktigt är även att ansvar och befogenheter tas och ges av personal inblandad i processen.

Ett sätt är även att placera personal i lokalen så att de är flexibla vad gäller vilket arbete som kan utföras och att flera aktiviteter kan utföras parallellt. Detta kan uppnås genom att använda den typ av organisation som projektet använder när det arbetar mot en deadline (Locher, 2011).

Pull

Principen Pull (sve. dra) syftar till att aktiviteter ska utföras just när de behövs samt att alla aktiviteter som utförs ska vara efterfrågade. Erfarenheter från tidigare komplexa projekt visar att antalet aktiviteter som inte behövs är signifikant (Oppenheim, Murman & Secor, 2009). Därför måste varje aktivitet i flödet behövas och efterfrågas av någon. Aktiviteterna bör därför väljas ut av projektledaren, i motsats till att aktiviteterna skickas ut av någon annan till projektet. Det innebär även att säkerställa att det inte finns onödiga köer, och att tillverka först när nästa steg kräver det för att undvika överproduktion och krav på lagringsplatser (Oppenheim, 2011). Genom sena och långsiktiga beslut kan snabbt genomförande förverkligas och de kortare leveranstiderna minskar effekten av förutsägelser som slår fel (Womack & Jones, 2003).

Pull är en metod för att kontrollera flödet av resurser baserat på verkliga behov och verklig konsumtion. I systemutveckling utgörs resurserna av information och människor, vad ska arbetas med, när och av vem, i syfte att undvika överproduktion. Informationsanhopning och tillgång till personal som arbetar med information är inte lika tydlig och lätt att synliggöra som när en typ av kablage anhopas vid produktionsplatsen (Locher, 2011).

Några av de möjliggörare som krävs för att kunna uppnå detta är (Oppenheim, 2011):

- Träna projektteamet till att kunna identifiera vem mottagare/beställare av en aktivitet är
- Tydliggör kraven för aktiviteten tillsammans med mottagare/beställare innan aktiviteten genomförs.
- Hålla kontakt med mottagaren/beställaren under aktivitetens genomförande.

Perfection

Principen Perfection (sve. perfektion) innebär att främja en kultur för successiva och ständiga förbättringar, vilket påverkar samtliga varianter av resursslöseri och ska genomsyra hela verksamheten (Womack & Jones, 2003).

Strävan mot perfektion innebär att hela tiden förbättra organisationens processer och kunskapsövertag, snarare än att ständigt förändra systemet eller kundvärdet. De främsta möjliggörare för att ständigt kunna förbättras är (Oppenheim, 2011):

- Utnyttja erfarenheter från tidigare projekt.
- Ha en kvalificerad chefsingenjör genom hela processen, dvs en chef med möjlighet att fördela befogenheter och utföra ändringar.
- Nyttja standarder för att befästa kunskaper som organisationen har erhållit.

Respect for people

Principen Respect for people (sve. respekt för människor) handlar om att bemöta kollegor (även chefer och underställda) i organisationen med respekt, och därigenom skapa ett engagemang för organisationen och projektet, med ett helt annat resultat än om medarbetarna inte känner sig värdesatta och därmed inte ger ”det lilla extra”.

I Lean förutsätts att anställda i organisationen har och ska ha egenmakt, samt vara tillgångar för organisationen. Systemutvecklare förutsätts sträva efter yrkesmässig utveckling. Miljön ska uppmuntra till lärande bland annat genom kontinuerlig utbildning och erfarenhet och genom synlighet av information (Oppenheim, 2011).

Enablers för att uppnå detta är bland annat (Oppenheim, 2011):

- Skapa en vision som inspirerar.
- Rekrytera kompetenta ledare och anställda.

- Uppmuntra respekt, ärlighet, relationsskapande aktiviteter och förtroende.
- Vårda en utvecklande miljö.
- Behandla människor som värdefulla tillgångar, inte som handelsvaror.

3.1.4.2 Toyotas 14 företagsledningsprinciper

Vad det gäller Lean är Toyota i stort ursprunget från vilket andra organisationer inte kan kopiera, men väl dra lärdom. I en serie av böcker har Toyotas förhållningssätt beskrivits, däribland de fjorton principer som Toyota förhåller sig till (Liker, 2004). Principerna är indelade i fyra kategorier: filosofi, process, människor och partners samt problemlösning.

Philosophy (sve. Filosofi) - långsiktigt tänkande:

1. Basera besluten på långsiktigt tänkande, även då det sker på bekostnad av kortsiktiga ekonomiska mål.

Process (sve.process)- eliminera slöseri:

1. Skapa processflöden som för upp problemen till ytan.
2. Låt efterfrågan styra, undvik överproduktion.
3. Jämna ut arbetsbelastningen.
4. Skapa en kultur där processen stoppas för att lösa problem det identifierade problemet, så att kvaliteten blir rätt från början.
5. Lägg standardiserat arbetssätt som grund till ständiga förbättringar och personalens delaktighet.
6. Använd visuell styrning, så att inga problem förblir dolda.
7. Använd bara pålitlig, väl utprövad teknik som stöder medarbetarna och processerna.

People and Partners (sve. människor och partners) - respektera, utmana och utveckla dem:

1. Utveckla ledare som verkligen förstår arbetet, lever efter företagets filosofi och lär ut den till andra.
2. Utveckla enastående människor och team som följer företagets filosofi.
3. Respektera det utökade nätverket av partners och leverantörer genom att utmana dem och hjälpa dem bli bättre.

Problem Solving (sve. problemlösning) - förbättring och lärande:

1. Gå och se med egna ögon för att verkligen förstå situationen.
2. Fatta beslut långsamt och i samförstånd, överväg noga samtliga alternativ, verkställ snabbt.
3. Bli en lärande organisation genom att oförtröttligt reflektera och ständigt förbättra.

3.1.5 Ledning av en Lean organisation

Ledning av en lean organisation bygger på att vara en ledare snarare än att vara en chef, att ge vägledning genom att vara en föregångare. En förutsättning för detta är att organisationen i grunden har anammat förhållningssättet. I Lean bygger ledning på (Locher, 2011):

- Driving continuous improvement (sve. driva ständig förbättring)
- Mentoring (sve. mentorskap)
- Going to the Gemba (motsvarande sve. leda på plats)
- Performance measurement (sve. prestationsmätning)
- Recognition (sve. erkännande)

För att få stöd i arbetet med ständig förbättring används W. Edward Deming's mer än 50 år gamla metodik Plan-Do-Check-Act (PDCA) där stort fokus läggs på att korrekt identifiera problemet innan orsakerna kartläggs (Arthur, 1992). Efter att lösning tagits fram implementeras och testas den, för att efter positivt testresultat slutligen föras in i standarder. Därefter lärs detta ut för att säkerställa att problemet inte uppstår igen. Metodiken är generell och säger ingenting om vilka metoder eller verktyg som bör användas för att planera ett test och analysera erfarenheter eller vilka generaliseringar som ska prövas i nästa test. Metodiken beskriver en cykel för kontinuerligt lärande och ständiga förbättringar, och som sådan gör den nytta över gränserna för olika projekt i organisationen. Organisationen leder genom att lära och vice versa.

Mentoring innebär att en ledare i en lean organisation måste betrakta sig själv som en lärare för sina underställda. I Lean innebär det att ofta öga mot öga dela med sig av information och förklara hur fattade beslut hänger ihop med ökat kundvärde. Det betyder inte att ledaren ska agera i ett klassrum full av elever.

Gemba är japanska för ”faktisk plats”. *Going to the Gemba* innebär att ledaren i utövande av ledning, så som diskussion av problem, ska vara på den plats där problemen finns.

Performance measurement innebär att ledaren kontrollerar att fattade beslut verkligen leder till ökat kundvärde och får därigenom bättre kontroll på sin process. Detta hänger mycket ihop med *Recognition* som innebär att ledaren ger positiv återkoppling till de anställda på utförd prestation i förhållande till ökat kundvärde.

3.1.6 Införande och potentiella fördelar

Att införa Lean sker i ett antal steg under en längre tid. Tar organisationen lärdom av andra organisationer kan ett införande förväntas ta minst 5 år. Utan andras erfarenhet tar det än längre tid. När tänkandet väl genomsyrar företaget väntar en fortsatt oavbruten utveckling (Locher, 2011). En modell i fem steg för införande i processindustri beskrivs enligt Womack och Jones (2003):

1. Under de första sex månaderna behöver en förändringsledare börja sprida kunskap om Lean, framförallt till chefer på ”golvet” och motivera varför metoden ska anammas. Är metodens fördelar inte uppenbara kan det vara lämpligt att provocera fram en kris för att få en tydligare bild av det nuvarande läget. Vidare kartläggs värdeströmmar.
2. Följande arton månader påbörjas omorganiseringen och fler chefer utbildas i Lean samtidigt som bakåtsträvare tas bort. Medarbetare informeras om att Lean kan ge personalflytt genom ökad tillväxt och inte genom avsked. I detta är det viktigt med respekt för de anställda.
3. Under år tre och fyra mäts och visas visuellt på metodens och därmed organisationens framgång, utan att hemlighålla resultat mellan organisationens delar. Belöning ges efter resultat och initiativ nerifrån premieras. Allmän utbildning genomförs.
4. Det avslutande av de fem åren fokuserar organisationen på att sprida Lean till angränsande organisationer som leverantörer. Fokus är på goda processer snarare än excellenta chefer och organisationen plattas till.
5. Efter de fem åren följer ständiga förbättringar och uppdateringar efter kundbehov.

De positiva konsekvenserna av införande inkluderar ökad produktivitet, snabbare processer och leveranser, minskade lagerbehov, högre kvalitet, färre personskador och ökad tillväxt genom frigjorda resurser (Womack & Jones, 2003).

3.1.7 Fallgropar

I detta avsnitt beskrivs ett antal fallgropar som gäller för införande eller vidmakthållande av Lean i en organisation.

3.1.7.1 Ofullständig implementering

Oavsett vilken förebild av Lean som används vid implementering i en organisation gäller att använda förhållningssättet konsekvent på alla nivåer i hela organisationen (Oppenheim, 2011). Att exempelvis enbart implementera de delar som motsvarar process-principerna i Toyotas 14 principer och avstå från övriga delar är en vanlig fallgrop för organisationer som bestämt sig för att använda Lean. Processdelen är inte tillräcklig för att hela förhållningssättet med dess produktionsfördelar ska uppstå i organisationen. Det är snarare så att då en organisation överarbetar processprinciperna och underutvecklar de övriga delarna uppstår en obalans i organisationen som kommer att stressa och utarma personal, leda till ett kortsiktigt tänkande och slutligen till att alla försök att arbeta efter Lean principer upphör (Locher, 2011; Liker, 2004;)

3.1.7.2 Respekt för människor

Respekt för människor glöms ofta bort i organisationer som försöker introducera Lean, varpå hela förhållningssättet fallerar. Respekt för samtliga personer som är involverade i systemutvecklingen är fundamentalt för Lean. Den innebär att ansträngningar måste göras för att skapa förståelse och ta ansvar för att bygga ömsesidigt förtroende i organisationen och med intressenter för att ett värde ska skapas. Lagarbete är en förutsättning för att skapa delaktighet i utvecklings- och förbättringsarbetet med syfte att maximera teamets och individens prestationer. Därmed stimuleras den personliga och yrkesmässiga utvecklingen, systemförståelsen och kvalitetstänkandet.

3.1.7.3 Kulturella skillnader

Lean beskriver ett förhållningssätt som bara fungerar fullt ut om vissa kulturella förutsättningar råder på samtliga nivåer i den systemutvecklande organisationen. De kulturella förutsättningarna utgörs av den ovan beskrivna synen på värde och slöseri, samt på de beskrivna principerna för att uppnå värdet och eliminera slöseri.

Lean är utvecklat i Japan, men ledningssystemet för Lean bygger på den typ av erfarenhetsbaserad lärandecykel som tillämpades runt 1920 i västvärlden. Den överfördes till Toyota på 1950-talet och utgör nu grunden för hela förhållningssättet. Det finns inget som säger att nationell kulturell tillhörighet skulle vara ett skäl för att Lean inte skulle kunna introduceras i ett företag. Det är snarare företagets interna kultur som är avgörande.

3.1.8 Reflektion

Förhållningssättet Lean har stora likheter med uppdragstaktik och i mångt och mycket tillämpas redan Lean-principer i Försvarsmakten. Vid införande av Lean är det ofta produktionsdelen av organisationen som prioriteras och blir lean,

medan ledning och anställda är lägre prioriterade i införandet. För Försvarsmakten är det annorlunda. Där finns vana att tillämpa flera av Leans principer på ledning och anställda, medan produktionsdelen är mindre prioriterad i det avseendet.

Detta kan ha att göra med att den PDCA-loop som fördes över till Japan på 1950-talet, och som ledning av en lean organisation bygger på, har stora likheter med den OODA-loop som John Boyd under Koreakriget började utveckla till en beslutscykel för militära operationer (Boyd, 1987). Erfarenhetsbaserat ledarskap, föregångsmannaskap, leda på plats, kontroll och återkoppling är ytterligare begrepp som förhållningssätten har gemensamt.

Försvarsmakten har förutsättningar för ett fullständigt införande av Lean, men skulle behöva anpassa ledning och produktion gemensamt till förhållningssättet. Det innebär bland annat att hantera ekonomiskt långsiktigt tänkande, öppenhet med information, systemförståelse och respekt för samtliga intressenter på alla nivåer.

3.2 Software Product Line Engineering

Software Product Line Engineering (SPLE) är en utvecklingsmetodik för mjukvara som bygger på en långtgående återanvändning av befintliga komponenter. En produktlinje definieras som en gemensam bas av tillgångar (eng. assets), vilka sätts samman till ett antal olika konfigurationer, där varje konfiguration tillgodoser ett visst behov inom ett marknadssegment. Det vill säga, SPLE är mer än bara utveckling av olika produktversioner då de resulterande produkterna kan skilja sig åt betydligt. Utanför mjukvarubranschen finns flera exempel på produktlinjer inom bil- och flygindustrierna samt snabbmatskedjor. Inom mjukvarubranschen är dock paradigmet relativt nytt, men organisationer med varierande verksamhet, organisation, kultur, mjukvarudisciplin och arvssystem har anammat det Software Engineering Institute(SEI) vid Carnegie Mellon-universitetet i USA har tagit fram ett ramverk för SPLE, där den senaste versionen påverkats av trender så som; (1) öppen källkod, (b) distribuerad utveckling, (c) Service-orienterad arkitektur, (d) modellbaserad utveckling och (e) agil utveckling (Northrop & Clements, 2007). Ramverket är tänkt att underlätta identifiering av koncept och aktiviteter för SPLE, samt att ge den specifika organisationen möjlighet att bedöma om en övergång till SPLE är möjlig. Nedan beskrivs SPLE baserat på SEI's ramverket.

3.2.1 Essentiella aktiviteter

SPLE baseras på tre grundläggande aktiviteter:

1. Skapandet av en bas med tillgångar som produkterna ska baseras på.
2. Produktutveckling.
3. Teknisk och organisatorisk administration för att genomföra de två föregående aktiviteterna.

Basen med tillgångar är mycket viktig och dess framtagande är såväl kritiskt som tidskrävande. Den måste tas fram med strategiskt tänkande där affärsperspektivet är minst lika viktigt som det tekniska. Samtidigt som basen skapas utformas även dokument som beskriver vilka produkter som ska produceras utifrån basen samt hur dessa ska produceras.

Inom klassisk mjukvaruutvecklingen återanvänds gärna kod, men detta utgör enbart en liten del av typiska projekt. I produktutveckling enligt SPLE plockas tillgångar ifrån basen och en ny produkt skräddarsys. Återanvändningen är betydligt mer omfattande, och placerat i basen finns även exempelvis större komponenter, krav, affärsmodeller, testfall, testdatamängder, dokumentation, arkitekturer, stödverktyg och domänmodeller. Varje gång en ny produkt utvecklas måste basen uppdateras. Således är arbetet iterativt med att utveckla produkter och skapa basen.

SPLE kan initieras antingen *proaktivt* eller *reaktivt*. I proaktiv SPLE skapas en första version av basen och sedan tas produkterna fram. Återanvändningen är därmed planerad istället för opportunistisk till skillnad från klassiska systemutvecklingsmetodiker. I reaktiv SPLE skapas basen utifrån befintliga produkter, vilket ger en lägre initial kostnad men också mindre allmängiltighet. Stoiber, Fricker, Jehle och Glinz (2010) har tagit fram en metod, benämnd *Feature unweaving*, för att från ett system skapa en produktlinje.

Att ta fram nya produkter genom utveckling ersätts till stor del av integrering och montering i SPLE, även om utveckling av nya komponenter ibland krävs. Viktigt är att inte skapa kopior av komponenter och utveckla dem åt var sitt håll utan att fokusera på inbyggd varians.

3.2.2 Tillämpningsområden

I ramverket finns tre tillämpningsområden som måste bemästras för att skapa en bra produktlinje:

- Tekniska mjukvaruutvecklingsaspekter, så som att skapa arkitekturer och komponenter, kravställa, testa och integrera.
- Tekniska ledningsfrågor så som utvärdering, mätning, planering, riskanalys samt verktygsstöd.

- Organisatoriska ledningsfrågor som relaterar till affärsmodellen, budgetering, marknadsanalys, organisationsutseende och utbildning.

I nästa avsnitt ges det första området särskild uppmärksamhet, medan de övriga två bedöms ha mindre att göra med SPLE:s grunder och mer att göra med generella systemutvecklingsmetoder. De är dock viktiga faktorer för de för- och nackdelar som följer.

3.2.3 Tillämpningsområden för mjukvaruutveckling

I detta avsnitt beskrivs olika mjukvarurelaterade aspekter för produktlinjer, med allt från arkitektur och komponenter till testning och integrering.

Arkitekturen är nyckeln till framgång för alla mjukvaruprojekt. Kritiska faktorer som styr arkitekturens utseende är hur specifika produkter tas fram från den generella arkitekturen, vilka affärsmässiga mål som finns och om komponenterna ska återvinnas, egenutvecklas eller köpas in. Att se till att önskad variation (det vill säga möjligt konfiguration av de olika produkterna) kan erhållas i arkitekturen är kritiskt, utan att komponenter varken blir alltför komplexa eller basala. Produkterna i produktlinjen kan skilja sig från varandra exempelvis vad gäller prestandakrav, kvalitet, skala, fysisk konfiguration, nät eller plattform. Specifika arkitekturer för varje enskild produkt tas fram så snart det är möjligt. Utifrån en lista över samtliga komponenter tas det beslut huruvida komponenten ska nyutvecklas, återvinnas från egna existerande system eller köpas in.

Kravhantering är en mycket viktig del inom SPLE. Kraven definierar produkterna i produktlinjen tillsammans med de olika produkternas egenskaper och begränsningar. Kravhantering inom SPLE skiljer sig från kravhantering för *ett* system främst genom:

- Kravinsamling – att fånga den förväntade variansen över tiden för serien av produkter sett till att det antagligen finns fler olika grupper av intressenter för produktlinjer än för system.
- Kravanalys – att analysera gemensamma och unika krav för de olika produkterna, identifiera kostnadsdrivande krav i de olika produkterna och vilka krav som är varianser i de olika produkterna. Sellier, Mannion och Mansell (2008) har tagit fram ett stöd i beslutshanteringen av vilka olika krav som tillsammans ska forma en produkt. Det finns många olika notationer för *feature modeling*, där gemensamma och varierande egenskaper i en produktlinje fångas i modeller. Några av dessa notationer är FODA, FORM och PLUSS (Alturki & Khedri, 2010). Inom målorienterad ansats (eng. Goal Oriented Requirements Engineering) kan metoden i* tillämpas för att identifiera gemensamma och varierande krav inom en produktlinje, genom ansatsen *Goals to Software Product Line* (G2SPL) (Silva, Borba och Castro, 2011).

- Kravdokumentation – att dokumentera de gemensamma och unika delarna för de olika produkterna, samt att hålla samman helheten.
- Verifiering av krav – innefattar först verifiering av den gemensamma basen för att sedan verifiera de olika produkternas egna krav.
- Förvaltning av krav – vid förändringar krävs det analys av både vad förändringen innebär för en enskild produkt, men även hur den påverkar de andra produkterna i linjen, och eventuellt basen.

Kim, Park och Sugumaran (2008) påtalar vikten av att ha spårbarhet mellan domänkraven och domänarkitekturen, för att produktlinjen ska vara flexibel avseende förändringar i behovsbilden. De har tagit fram en metod, DRAMA, för att modellera domänarkitekturen baserat på domänkraven, med syfte att smidigt hantera förändringar i behoven från marknaden.

Tester syftar till att identifiera och eliminera fel samt säkerställa att systemet uppfyller kraven. Det finns ett stort antal tester som bör utföras dels på basen och dels på respektive produkt.

För att lyckas med en produkt är det av yttersta vikt att förstå den eller de domäner som produkterna förväntas fungera i. Inom SPLE innebär det, jämfört med att utveckla ett system, att inte bara förstå domänen, utan även de olika intressenternas behov och hur de grupperas till lämpliga produkter.

3.2.4 Fördelar

SPLE kan vara en ekonomiskt fördelaktig ansats där ett flertal produkter kan skapas till relativt sett låga kostnader vad gäller underhåll och tillverkning. Fördelar med SPLE innefattar:

- Ökad produktivitet och snabbare utveckling genom att nyttja basen för återanvändning.
- Ökad kvalitet då varje nytt system utnyttjar elimineringen av fel i tidigare versioner.
- Ökad kundnöjdhet, bland annat genom förutsägbara leveransdatum och kostnader, vältestat utbildningsmaterial och dokument, delad underhållskostnad och möjlighet att dela erfarenheter.
- Användare lär sig en produktlinje istället för fristående produkter och för slutanvändare finns fler användare att rådfråga.
- Minskad risk per produkt eftersom det finns erfarenheter att falla tillbaka på och varje produkt enbart innehåller liten del experimentellt risktagande.

- Lägre krav på programmeringskunnande genom minskat behov av utveckling
- Reducerade utvecklingskostnader och time-to-market (Sellier, Mannion och Mansell, 2008).

3.2.5 Nackdelar

Det har även identifierats nackdelar med SPLE avseende tekniska, organisatoriska och ledningsmässiga risker:

- Det är riskfyllt att introducera ett nytt produktionssätt. Att hitta komponenter och att integrera tar tid och komponentdokumentation som finns är inte generell nog.
- Strategiskt kapital och organisatorisk mogenhet krävs.
- En visionär ledare krävs för att driva en produktlinje.
- Nya anställda måste läsa in sig på hela produktlinjen förutom att lära sig SPLE, samtidigt som arvssystem som kräver äldre utvecklingsmetodik ställer krav på traditionella kompetenser i organisationen.
- Dokumentation och spridning av kunskap är viktig eftersom en medarbetare kan vara inblandad i ett steg av en enskild produkt i produktlinjen och när ett överlämnande till nästa steg sker krävs att medarbetarens kunskap är dokumenterad för att nästa steg ska fortlöpa så väl som möjligt.
- Ökade krav på domänkunskap och tekniska förutsägelser samt strategiskt arbete.
- Ökad uppstartskostnad och dessutom krävs underhåll av basen.
- Bristfälliga stödverktyg för den nya metodiken.
- Det är svårt att hålla sig tillräckligt generell för att tillåta varians samtidigt som varje produkt ska bli tillräckligt specifik.
- Många produkter står och faller med basen.

3.2.6 Reflektioner

Inom Försvarsmaktens utveckling av ledningssystem finns en ambition att återanvända utvecklade funktioner som är gemensamma för olika typer av ledningssystem och att korta utvecklingscyklarna för ledningssystem. SPLE åstadkommer detta inom mjukvaruområdet och skulle kunna generaliseras för att fungera inom utveckling av ledningssystem.

3.3 Six Sigma

Motorola utvecklade Six Sigma i slutet av 1980-talet för att mäta företagets kvalitet i termer av antal defekter. Det ansågs på företaget att möjlighet att mäta innebar möjlighet att påverka. Definitionen av defekter var bred, vad som helst som inte lever upp till kundförväntan eller till ställda krav. Räkning av defekter applicerades på alla processer i företaget i syfte att eliminera defekterna och därmed nå processer med mindre variation.

3.3.1 Användningsnivåer

Six Sigma har utvecklats under de tjugo år som passerats och meningarna går isär om vad Six Sigma är idag. Six Sigmas ursprungliga syfte var att förbättra redan befintliga processer. Motorola, de ursprungliga utvecklarna, anser att Six Sigma kan vara en metrik, en metod och ett förhållningssätt vilket även är de tre vanligaste tolkningarna av Six Sigma (Ficalora & Cohen, 2010).

3.3.1.1 Metrik

Metriken Six Sigma används för att räkna defekter i en organisation. Med defekt menas vad som helst som inte lever upp till ställda krav. Defekter kan finnas inom vilket område som helst i organisationen. Sigma används som en skala för nivå av kvalitet och just Six Sigma innebär en nivå av 3,4 defekter per miljon möjligheter i organisationen som helhet. Detta följer av att sigma står för standardavvikelse, en sorts spridningsmått. Sex standardavvikelser motsvarar för en normalfördelning just 3,4 på 1 miljon. Varför just sex standardavvikelser har valts är inte helt klart. Normalt är man dock flexibel med den exakta siffran och använder istället den generella ansatsen. I de fall då Six Sigma används enbart som en metrik kan en obalans uppstå i organisationen. Då ingen prioritering mellan defekterna görs läggs lika mycket möda på oväsentliga defekter som på väsentliga. Detta är ett av de spörsmål som ofta kritiseras vid användning av Six Sigma där organisationen mäter men mäter fel saker (Ficalora & Cohen, 2010).

3.3.1.2 Metod

Vartefter Six Sigma utvecklats har betoningen på den bokstavliga definitionen 3,4 defekter per miljon möjligheter minskat. I stället ses Six Sigma som en förbättringsmetod för organisationer som bättre vill hantera kundens krav genom att med hjälp av statistiska metoder anpassa processer för att minimera processernas variationer och därigenom erhålla varaktiga processer där kvaliteten är bestående. Att arbeta med Six Sigma som en projektmetod innebär att mätområdena inom projektet prioriteras. Enbart defekter som väsentligt påverkar kundförväntan eller ställda krav och som ingår i projektplanen mäts och åtgärdas. Metoden Six Sigma innebär ett arbete med defekter enligt Definiera, Mäta,

Analysera, Förbättra, och Kontrollera effekterna. Detta förkortas vanligen DMAIC efter de engelska motsvarigheterna, *Define*, *Measure*, *Analyze*, *Improve* och *Control* (Ficalora & Cohen, 2010).

Definiera innebär att ta fram en plan där hänsyn tas till krav som motsvarar kundens förväntningar, till projektets avgräsningar och till de ekonomiska förutsättningarna. Den investering en Six Sigma-insats utgör i projektet ska vara lönsam. Den process som ska kvalitetssäkras enligt Six Sigma kartläggs i detta steg.

Mäta innebär att samla mätdata för att skapa en baskunskap om processen och hur den fungerar innan kvalitetsarbetet. Arbetet innebär inte att mäta allt, utan att identifiera och mäta aktiviteter i processen som har ekonomisk och kvalitativ betydelse för processens utfall (eng. characteristics that are critical to quality)

Vid *analys* är syftet att finna de faktorer som påverkar produktens kvalitet. När faktorerna är funna kan arbetet med att styra processen mot bästa design börja.

Vid *förbättring* av processen enligt de förutsättningar som funnits vid analysen ska förändringen inte bara genomföras utan även följas upp och verifieras.

Kontroll är det tillstånd processen kommer att befinna sig i efter att de stora insatserna har genomförts. Kontroll innebär att vidmakthålla förbättringen genom styrning och uppföljning.

När Six Sigma används som en projektmetod ses DMAIC ofta som en linjär process med en början, ett utförande och ett slut. Syftet med DMAIC är att förbättra en process och bevisa att processen förbättras. DMAIC tillför effektivitet i projektet på just den punkten den är avsedd för, men den har inte till syfte att öka organisationens lärande eller ge synergieffekt på andra processer.

3.3.1.3 Förhållningssätt

Då Six Sigma anammas som förhållningssätt i ett helt företag läggs ytterligare dimensioner till DMAIC. Varje delsteg i DMAIC är klarare och har en tydlig vision i förhållande till hela organisationens kvalitetsarbete. DMAIC övergår från att vara linjär till att bli en cirkulär process. Då Six Sigma är en top-down-ansats innebär det att högsta ledningen i företaget måste vara mycket delaktig i arbetet. Ledningen behöver utarbeta en vision för företaget som är möjlig för enheter på alla nivåer att avdela mätbara mål ifrån. De behöver ha långsiktig förmåga att skapa acceptans och delaktighet för förhållningssättet som i stort bygger på DMAIC runt de mätbara målen. Ledningen utbildar systematiskt personal för att hantera arbetssättet samt förser personalen med tillräckliga verktyg. I Six Sigmas utbildningssystem ges olika individer i företaget olika roller med tydligt ansvar och mandat (Ficalora & Cohen, 2010). I traditionellt utförande är dessa roller benämnda liknande ett graderingssystem för en tänkt asiatisk kampsport med bland annat Champions, Black Belts, Green Belts. Den utbildning som motsvarar

gradsystemet är starkt fokuserad på tankarna runt Six Sigma med mätning, verktyg och förståelse för varje steg i DMAIC.

Graderna beskrivs som (Penn & Siviy, 2003):

Executive Leadership: Sätter upp övergripande mål och ordnar så att resurser finns för att lyckas med projekt.

Champions: Har det övergripande ansvaret i organisationen och är mentorer för Black Belts.

Master Black Belts: Ägnar sig åt Six Sigma på heltid, utbildar black belts och är experter på metodik och statistik. Master Black följer projekten och ser ut lämpliga nya projekt.

Black Belts: Leder på heltid förbättringsprojekt i organisationen. Dessa är experter på att leda projekt enligt DMAIC och ger tydliga besparingar till företaget. Utbildning till Black Belt innehåller bland annat projektledning och avancerade statistiska metoder.

Green Belts: Arbetar deltid med Six Sigma uppdrag och både leder och arbetar i projekt. Green Belts har bland annat utbildning i statistiska metoder.

Yellow Belts och *White Belts*: Är grader från utbildningar med syftet att få baskunskap för att delta i förbättringsprojektgrupp eller att vara en del av en avdelning eller sammanhang där ett Six Sigma projekt pågår.

Förhållningssättet Six Sigma är i och med den speciella utbildningsansatsen expertberoende och ledning av ett Six Sigma projekt bygger på att chefen har en specialkompetens. Chefen förmedlar vad, hur och när till sina anställda utefter en plan som sannolikt inte är känd för den anställde. De anställda har inte utbildats och bredvidliggande processer har inte berörts. Synergieffekter i organisationen förutsätts inte.

3.3.2 Design for Six Sigma

En utmaning för förhållningssättet Six Sigma har varit att omhänderta kreativitet i syfte att utveckla nya processer och produkter. Vanligast är att Six Sigma förbättrar befintliga processer och produkter, men att nya sådana inte tas fram (Mader, 2002). För att anta utmaningen har en parallell process till DMAIC utvecklats. I sitt grundutförande benämns processen DMADV där de avslutande D och V oftast står för *Design* och *Validera*. Tanken är att förändringen i de två sista stegen ska borga för ny design av processer och produkter. Förändringen av det ursprungliga kvalitetsstyrningsmetoden mot design kallas med ett gemensamt namn *Design for Six Sigma* (DFSS). Det finns dock en uppsjö av varianter av DFSS:er där DMADV bara var den första. De allra flesta DFSS:er verkar inte kunna stå på egna ben utan förlitar i mycket sig på att organisationen redan har en produktutvecklingsprocess (Ficalora & Cohen, 2010). I stället har DFSS:erna

antagit en form där de kvalitetssäkrar företagets egna systemutvecklingsprocess med grund i samma tankar om mätning, analys och kvalitetskontroller som ingår i traditionell Six Sigma (Mader, 2002).

3.3.3 Fallgropar

Grundproblemet för Six Sigma är att i en komplex miljö är att det inte finns tillräckligt med chefer i förhållande till medarbetare och övriga intressenter. Detta är en vanlig fallgrop i top-down ansatser. I en top-down ansats räcker ledningen inte till för att detektera och åtgärda samtliga problem. Den stora massan av intressenter är de som vet att problemen finns. Dessa måste ha en kanal för att förmedla sin information till ledningen. Six Sigma i sig borgar inte för den typen av kommunikation. Då uppstår ett läge där ledningen döljer information för intressenter i allmänhet, och där intressenter inte har förmåga att förmedla sin information till ledningen (Oppeheim, 2011, s 202).

Den gradbaserade utbildningen i förhållningssättet är inte kopplad till erfarenhet, vilket innebär att DMAIC-lärlingar kan väljas efter begåvning för statistisk analys snarare än erfarenhet av organisationens kunder, processer och produkter. I mötet mellan personal som representerar erfarenhet och Six Sigma-lärlingar kan enkelt kommunikationsproblem uppstå, speciellt i de fall där Six Sigma-initiativet hotar erfaren personals förutsättningar i processen (Liker, 2011)

Förhållningssättet Six Sigma är i och med den speciella utbildningsansatsen mer expertberoende än Lean som grundas i en bred bas av utbildade och motiverade anställda. Six Sigma har en top-down-ansats medan Lean arbetar i alla nivåer. Förhållningssättet har fått kritik ur ett Lean-perspektiv för att slösa med resurser genom att åtgärda problem som egentligen kräver lågaktiva åtgärder. Metoden Six Sigma ska användas som ett verktyg för att eliminera varians enligt lean-principen flöde vid stora problem där förståelsen för problemet kräver statistisk dataanalys (Oppenheim, 2011).

3.3.4 Reflektion

Six Sigma är en metrik, metod, eller ett förhållningssätt som syftar till att utveckla processer till en mycket hög kvalitetsnivå genom expertkunskap och rigorös statistisk analys. Den typen av kvalitetsarbete utgör troligtvis inte ett behov för Försvarsmakten generellt. Vid specifika utvecklingsaktiviteter kan dock Försvarsmakten ha nytta av den typen av kvalitetsarbete.

3.4 Capability Maturity Model Integration

Capability Maturity Model Integration (CMMI) är en samling metoder för processförbättring och ett sätt att bedöma processer för att identifiera svagheter. CMMI kan underlätta vid arbete i stora och komplexa systemutvecklingsprojekt genom att ta ett holistiskt perspektiv som eliminerar stuprör och andra hinder för effektiva processer. Med CMMI underlättas processförbättring i organisationer, dock anges inte hur processer ska se ut i detalj då variationer måste förekomma mellan organisationer. Det finns flera varianter av CMMI och vilken som bör användas beror på vilka typer av projekt den specifika organisationen driver. I denna rapport fokuseras främst på CMMI för utvecklingsprojekt och där inte annat anges är därför källan *CMMI® for Development* (CMMI Product Team, 2010).

CMMI har sina rötter i Shewharts (1939) kvantitativa kvalitetskontrollsarbete på 30-talet och togs fram på slutet av 80-talet inom IBM och Carnegie-Mellons Software Engineering Institute (SEI). SEI äger nu bokstavligen begreppet och driver utvecklingen av CMMI. CMMI bygger på flera tidigare så kallade Capability Maturity Models och var från början mjukvaruorienterat, men används idag även för utveckling inom andra områden såsom finans- och bilindustri.

3.4.1 Processområden

CMMI delar in processer i ett antal processområden vilket består av ett antal specifika och generiska mål samt de aktiviteter som behövs för att nå målen. Generiska mål gäller för alla processområden och består av sådant som: planering, resurssäkring, utbildning, organisation, kontroll och utvärdering samt involverande av relevanta intressenter.

Processområden för utvecklingsprojekt inom CMMI är dels 16 processområden som är gemensamma med andra CMMI-modeller, dels ytterligare 6 processområden som är mer specifika. I Tabell 2 följer en komplett lista på dessa processområden, tillsammans med tillhörande mognadsnivå (som diskuteras senare).

Tabell 2 Tabellen visar CMMI:s processområden med tillhörande mognadsnivå.

Processområde	Mognadsnivå
Ändringshantering	2
Mätning och analys	2
Projektövervakning och kontroll	2
Projektplanering	2
Process- och produktkvalitetssäkring	2
Kravhantering	2
Leverantörsavtalshantering	2
Beslutsanalys	3
Integrerad projekthantering	3
Organisatorisk processdefinition	3
Organisatoriskt processfokus	3
Organisatorisk utbildning	3
Produktintegration	3
Kravutveckling	3
Riskhantering	3
Teknisk lösning	3
Validering	3
Verifiering	3
Organisatorisk processprestanda	4
Kvantitativ projekthantering	4
Kausal analys	5
Organisatorisk prestandahantering	5

3.4.2 Mognads- och förmågenivå

Mognadsnivå samt förmågenivå beskriver hur en organisations processer förhåller sig till CMMI. Förmågenivån bedömer organisationens förmåga för ett enskilt processområde på skalan 0 till 3, medan mognadsnivåer uppskattar mogenheten för en mängd processområden eller en hel organisation i intervallet 1-5. Mognadsnivån är en standardisering som gjorts på processområden och påtalar vilka processområden en organisation måste fokusera på för att nå en viss mognadsnivå. Bedömningar efter förmågenivån ger utrymme för organisationen att själv välja vilka processområden som är viktiga att förbättra i organisationen.

3.4.2.1 Förmågenivå

Förmågenivån är en skala 0-3 där organisationens utvalda processområden bedöms enligt:

0. Inkomplett – på denna nivå är processer inkompletta eller saknas helt.
1. Utförd – på denna nivå ges indikationer på att specifika mål nås för processområdet, men det är oklart om denna status kommer att bibehållas i organisationen.
2. Hanterad – på denna nivå innebär det att det finns en policy för hur processer ska utföras även om ingen organisationsomfattande standard existerar. Dessutom involveras intressenter och processerna kontrolleras för korrekthet mot processbeskrivning.
3. Definierad – på denna nivå är processer definierade med procedurer och standarder, som gäller för samtliga projekt i hela organisationen.

När en organisation antar CMMI och ska genomgå förbättringar gällande förmågenivån är det vanligt att en målprofil skapas. Målprofilen anger de processområden som är tänkta att förbättras samt med hur mycket. Här måste även de inbördes beroendena mellan olika processområden hanteras. Dessa beroenden kan göra att det inte går att förbättra enbart ett specifikt processområden, utan att förbättra även andra. Hur olika processområden hänger ihop återfinns i SEI:s standarddokument om CMMI (CMMI Product Team, 2010). Att förbättra sig utöver förmågenivå 3 innebär att satsa mot de högre mognadsnivåerna (4 och 5) som beskrivs mer detaljerat nedan. Då kan organisationen nå högre prestanda genom kvalitativa metoder.

3.4.2.2 Mognadsnivå

Mognadsnivån bedömer en mängd processområden i en organisation och uppskattar organisationens mognad men bedömer inte varje enskilt processområde djupare utan enbart på en högre nivå. För varje mognadsnivå finns ett antal förutbestämda processområden som ska beaktas. Detta står i kontrast till förmågenivåerna där organisationen själv väljer vilka processområden som ska vara i fokus. Medan förmågenivåerna bättre kan spegla den specifika organisationen, underlättar mognadsnivåer vad gäller kopplingar mellan olika områden och i jämförelse mellan organisationer. Skalan för mognadsnivån är 1-5 och bedöms enligt:

1. Initial – på denna nivå är processer ofta framtagna på projektbasis och eventuella framgångar beror på individuella prestationer. Arbetssättet är adhoc och innebär att upprepad framgång är svåruppnådd. Ofta överskrids budget, både vad gäller tid och pengar.
2. Hanterad – på denna nivå präglas organisationen av kontrollerade processer som utförs enligt policy och plan. Intressenter involveras och

milstolpar avrapporteras. Processer frångås inte på grund av stressade situationer.

3. Definierad – på denna nivå är processer välförstådda och är spridda i hela organisationen. Detta medför att processer ser likadana ut mellan projekt med undantag för mindre anpassningar. Processbeskrivningar är dessutom mer rigorösa med bland annat beskrivna aktiviteter, roller, verifieringssteg samt utdata och utgångskriterier.
4. Kvantitativt hanterad – på denna nivå inkluderas kvantitativa målkriterier som baserar sig på kund- och användarnytta. Mer statistiskt förankrade och tillförlitliga mätningar utförs och i vissa väl valda fall utförs mätningar på underprocessnivå för mer exakthet.
5. Optimering – på denna nivå genomförs ständig processförbättring och statistiska metoder underlättar förståelse för processvarians. För att spegla teknik- och organisationsförändringar förbättras successivt processer och jämfört med nivå 4 skaffas en organisatorisk helhetsbild.

Bedömd mognadsnivå kan användas för att förutsäga hur väl projekt kommer att fortlöpa, samt deras resultat eftersom dessa aspekter är proportionerliga med den organisatoriska mognaden.

Enligt Process Maturity Profile (2010) är mognadsnivå 3 klart vanligast, men för organisationer under 300 anställda är nivå 2 även välrepresenterad och 85 % av mätta företag är på mognadsnivå 2 eller 3. Mognadsnivå 5 är sällsynt och främst vissa större organisationer återfinns där. Att gå ifrån mognadsnivå 1 till mognadsnivå 5 tar som median sex år. Att försöka hoppa över mognadsnivåer är mycket riskfyllt då brister ofta visar sig när organisationen hamnar i en stressfylld situation. Att implementera en del av exempelvis mognadsnivå 4 när resten av organisationen befinner sig på mognadsnivå 2 är ofta bortkastat, detta på grund av att tolkningsmöjligheter saknas för datavärden eftersom mätmetoden ligger efter.

3.4.3 Införande av CMMI i organisationen

Vid införande av CMMI i en organisation bör följande beaktas:

- En stark ledare högt upp i organisationen krävs för att få stöd för förändringarna och för att säkra resurser. Denne ledare väljer ut starka medarbetare som kan leda delar av arbetet i organisationen. Dock sker uppföljning av arbetet kontinuerligt av ledaren.
- En tekniskt kompetent processgrupp bör finnas för att se till att tekniska frågor för organisationens olika delar beaktas.

- En del av organisationen väljs ut för att begränsa införandet av CMMI. Detta medför en hanterbar förändringsprocess i början, för att senare sprida förnyelsen till den resterande organisationen.
- Förmågenivå eller mognadsnivå bör väljas, mot vilka uppsatta mål mäts. Utifrån detta väljs vilka processområden som först ska bearbetas, varpå hänsyn till processområdenas beroende måste beaktas. Organisationens nuvarande processer måste även översättas till indelningen av processområden som görs i CMMI.
- Medarbetare behöver utbildas i det nya tänket.
- Organisationen behöver besluta om huruvida processförbättringen ska mätas och vilken metod och skala som ska användas.

3.4.4 Mätning av mognad och förmåga

Det finns flera orsaker till att mäta sin nuvarande förmågenivå eller mognadsnivå:

- Vetskap om vilka förbättringar som kan genomföras.
- Uppfyllande av kontraktskrav från kund eller marknadsföring av organisationen.
- Jämförelse med andra organisationer.

En förmågenivå för ett viss processområde talar om hur kompetent organisationen är på det processområdet. En mognadsnivå talar istället om hur kompetent organisationen är på en specificerad samling processområden. Om organisationen är extra duktig på ett enskilt processområde som ingår i mognadsnivån, premieras inte detta. Detta gör att det, för en organisation som tidigare använt sig av förmågenivåer, inte är helt rättvisande att översätta från förmågenivåer till mognadsnivåer. Vidare är det inte helt lätt att jämföra enskilda processområdens förmågenivåer mellan organisationer, bland annat eftersom inga standardiserade mätningar för enskilda processområden finns. Trots detta kan förmågenivån vara av intresse för organisationer. Att använda förmågenivån baseras på att lägga extra fokus på lämpliga processområden för organisationen, istället för att gå standardspåret för att uppfylla mognadsnivåerna.

Mätningar av mognadsnivån kan göras olika formellt, på en skala A-C där A är mest formell. Mer informella metoder baserar sig på att organisationen bedömer sitt eget arbete internt eller att en snabbare kontroll genomförs. Den vanligaste metoden för formella bedömningar är SCAMPI A (SCAMPI Upgrade Team, 2011), och det är den enda metod som lämpar sig för att jämföra sig med andra organisationer. SCAMPI A innebär en mätning av organisationen av en extern grupp, certifierad av SEI. Istället för att genomföra en mätning på hela

organisationen kan en del av organisationen väljas ut för mätning, vilket resulterar i ett snabbare genomförande och att det går att fokusera på vad som upplevs som den kritiska delen av organisationen.

Resultatet av en formell bedömning kan, om organisationen vill, publiceras hos SEI i deras lista PARS. I listan märks flera stora och välkända företag såsom IBM, Toshiba, Siemens, samt svenska Viasat (PARS, 2011).

Att låta en extern grupp mäta organisationens processer kan verka riskfyllt men SCAMPI A trycker på sekretesstänket, exempelvis gällande hantering av känsliga uppgifter vid arbetsgången och vad av resultatet som redovisas för SEI.

3.4.5 CMMI och relaterade systemutvecklingsmetoder

CMMI kan underlätta att ta reda på vilka processer som behöver förbättras men hjälper inte till med hur. Därmed ställer sig CMMI neutralt till vilken utvecklingsmetod som används.

3.4.5.1 Six Sigma och Lean

Att använda Six Sigma tillsammans med CMMI är en ganska vanlig kombination. Six Sigma fokuserar på kvalitetskritiska faktorer i hela organisationen och kan halvera tiden det tar att nå en högre mognadsnivå i CMMI. Metoden Six Sigma baserar sig likt CMMI på mätningar i organisationen (CMMI Appraisal Program, 2010). Six Sigma kan upptäcka behov av bättre processer (Siviy, Penn & Harper, 2005) och tillsammans med förmågenivån i CMMI kan de för kvalitetskritiska processområdena identifieras (CMMI Appraisal Program, 2010). Arbetet inom Six Sigma underlättas då en större processmognad finns i organisationen. Flera organisationer har lyckats kombinera Six Sigma och CMMI, till exempel vapentillverkaren Raytheon. Även Motorola, som ligger bakom Six Sigma, har lyckats kombinera Six Sigma och CMMI (Gibson, Goldenson & Kost, 2006).

Lean kan användas i samband med CMMI, med eller utan Six Sigma. Mätningarna i CMMI ger ett bättre och mer objektivt underlag för att fatta beslut i Lean och flera processområden kan ge direkta fördelar för Lean. Exempelvis involverar processområdet validering kunden, medan processområdet kausal analys underlättar slöserireducering (Siviy, Penn & Harper, 2005). År 2002 nyttjade ABB Lean i kombination med CMMI för kravprocessförbättringar (Gibson, Goldenson & Kost, 2006).

3.4.5.2 Agila metoder

Medan CMMI säger vad som ska göras, fokuserar agila metoder snarare på hur det ska göras. Därmed föreligger ingen inneboende motsättning mellan dessa. Agila metoder fokuserar mycket på individer och projekt framför process och organisation, på att skriva mjukvarukod före dokumentation samt på flexibilitet

framför hårda och fasta planer. CMMI har därmed ett längre tidsperspektiv och mäter framgång mer genom vilka processer som finns än vad som åstadkoms. Agila metoder är mer intresserad av att göra det bästa av det man har nu, snarare än att fokusera på att nå nya nivåer av mogenhet eller förmåga (Glazer, Dalton, Anderson, Konrad & Shrum, 2008).

CMMI har historiskt använts för stora och kritiska projekt, medan agila metoder företrädesvis nyttjats för små mjukvaruprojekt. Detta har gett en uppfattning att metoderna enbart passar dessa specifika världar. Dessutom är det en felaktig föreställning att högre CMMI-nivåer kräver att processerna är återupprepnings-bara förekommit. Repeterbarhet var enbart gällande CMMI:s föregångare CMM (Glazer et al., 2008). Traditionellt har det ansetts svårare att kombinera agila metoder med CMMI på mognadsnivå 4 eller 5, men till och med mognadsnivån 5 har använts i samband med Scrum och Lean istället för vattenfallsmodellen (Jakobsen & Sutherland, 2009). Inbyggt i SEI finns ett antal råd för hur vissa CMMI-processser lämpligast används i kombination med agila metoder och deras fokus på användarinvolvering, samt iterering.

3.4.6 Reflektioner

Även om CMMI inte är så stort i Europa, än så länge, kan det noteras att i USA dominerar försvarsmaktsrelaterade organisationer användandet (Process Maturity Profile, 2010). Vidare har SEI samarbetat mycket med amerikanska försvarsdepartementet. I Sverige är det ett fåtal organisationer som idag rapporterat till SEI, däribland kan Viasat återfinnas. Vid användandet av CMMI är en viktig fallgrop att helt förlita sig på att CMMI ska mäta och upptäcka problem, istället för att förebygga dessa på ett tidigare stadium (Glazer et al., 2008).

3.5 Goal-Oriented Requirements Engineering

För att utveckla effektiva system är det viktigt att identifiera och specificera de behov som systemet ska uppfylla, samt vad systemet ska utföra. Behovet av att beskriva *varför* system skulle byggas identifierades tidigt inom kravhanterings-området, men fick inte så mycket uppmärksamhet jämfört andra aspekter. Fokus låg istället på *vad* systemen skulle utföra och *hur* detta skulle ske. Detta medförde att det inte togs tillräcklig hänsyn till den omgivning i vilket systemet skulle användas och att skapa insikt i intressenternas *verkliga behov* prioriterades inte (Lapouchnian, 2005). Enligt van Lamsweerde (2009) kan krav hanteras utifrån tre aspekter – *varför* behövs det framtida systemet, *vad* för behov ska det hantera och *vilka* deltar för att systemet ska uppnå sina mål? En ansats för att reda ut dessa frågor är att utgå från vilka *mål* intressenterna har och vad de ska uppnå. Denna ansats benämns som Goal-Oriented Requirements Engineering (GORE).

Det system som ska utvecklas och dess kontext (eng. environment) ses inom GORE som en samling av aktiva komponenter, benämnda som agenter (eng. agents). Agenterna kan vara människor, utrustning och mjukvaror vilka kan förändra sitt beteende för att uppfylla kriterier de tilldelats. De ansvarar även för att uppfylla systemets mål (Lapouchnian, 2005). Inom GORE ses organisationers och intressenternas mål som källan till så väl funktionella så som icke-funktionella krav. Av denna anledning måste deras mål identifieras innan kraven kan identifieras. Detta skiljer sig från flera traditionella ansatser till krav-hantering, som präglats av produktfokus med bristfällig återkoppling från användare och verksamhet. Detta ledde till att kraven på systemen inte svarade mot verksamhetens och användarnas reella behov. Då fokus låg på systemkrav och inte de verkliga behoven medförde detta problem som avsaknad av spårbarhet mellan behov, krav och design samt kravdokumentation som var inkonsekvent eller ofullständig (Hallberg, Haraldsson, Lewau et al., 2011). Genom att istället fokusera på målen som varje intressent i ett system har och resonera kring vem och varför söker GORE att förbättra kravhanteringen (van Lamsweerde, 2011).

Det finns en rad olika ansatsers som alla kan klassificeras som att tillhöra GORE (Werneck, Oliveria & Leite, 2009) där de två mest omskrivna metoderna är i* och KAOS. Baserade på i* har det även utvecklats ytterligare varianter som utökat ramverket det på olika sätt och här kan framförallt TROPOS och GLR nämnas. Dessa tillsammans med i* fokuserar på systemets intressenter och kan benämnas som *Agent-Oriented Requriement Engineering* (AOSE). Enligt van Lamsweerde (2009) är dock GORE och AOSE i grunden lika och skiljer sig endast åt gällande arbetssättet. AOSE utgår från systemets agenter och deras relationer, för att från detta identifiera systemets mål. Hos KAOS identifieras först systemets mål och utifrån dessa mål identifieras berörda agenter. Skillnaden i fokus innebär även att metoderna skiljer sig hur detaljerat antingen agenter eller mål dokumenteras (Werneck, Oliveria & Leite, 2009).

Den här rapporten kommer härefter att ge en översiktlig beskrivning av GORE utifrån KAOS-ramverket beskrivet i van Lamsweerde (2011). För att kontrastera mot detta ges därefter kortare beskrivningar av i* samt NFR-ramverket (Fricker & Glinz, 2010; Lapouchnian 2005).

3.5.1 KAOS

Grundprincipen i KAOS är att identifiera varför ett system behövs, vilka som är involverade och hur system ska uppnå sina mål. Detta görs genom att först identifiera vilka mål systemet har och successivt dela upp mål i allt högre detaljgrad. Genom detta arbete identifieras då även vilka agenter som är berörda av systemet samt vilka mål de antingen har själva eller delar med andra. Att bygga upp kravhanteringen kring systemets mål anser van Lamsweerede (2009)

ger ett naturligt sätt att strukturera de komplexa förutsättningar som föreligger systemet.

3.5.1.1 Preskriptiva och deskriptiva utsagor

Ett central begrepp inom samtliga olika ansatser inom GORE är termen mål. Ett mål definieras här som ett tillstånd som systemet har förmåga att reglera eller styra över (van Lamsweerde, 2009). Ett mål kan till exempel vara *Möten ska schemaläggas så att deltagandet av inbjuda maximeras.* Målet uttalar en intention som ska uppfyllas av systemets agenter, i detta fall ett planeringsverktyg, mötesdeltagare och mötesinbjudare. Detta kallas även för en preskriptiv utsaga då det säger något som ska uppfyllas. I Tabell 3ges en översikt av de tre typer av preskriptiva utsagor som finns inom KAOS.

Varje mål kan befinna sig på olika abstraktionsnivåer, där exemplet ovan är förhållandevis abstrakt och uttalar nästan en strategisk inriktning. Ett mål på en lägre abstraktionsnivå, så som *Endast behöriga ska ha möjlighet att initiera möten,* ligger närmre en teknisk inriktning med olika designalternativ. Vilken abstraktionsnivå eller detaljgrad ett mål befinner sig på är kopplat till antalet agenter som är involverade för att uppnå målet. I det första målet krävs samverkan mellan de tre agenterna: planeringsverktyget, personen som bjuder in till mötet samt de som bjuds in. Det andra målet däremot berör endast två agenter nämligen planeringsverktyget och den initierande parten.

Tabell 3 Typer av preskriptiva utsagor

Term	Beskrivning
Mål	Systemtillstånd som ska uppnås genom samverkan av en eller flera agent
Krav	Mål som kontrolleras av en agent, och där måluppfyllnad kan framtvingas av systemet.
Förväntan	Mål som kontrolleras av en agent i omgivningen, där måluppfyllnad inte kan framtvingas av systemet.

Genom att ytterligare detaljera målen bryts dessa successivt ner tills endast *en* agent är involverad. Exempelvis skulle då ett mål kunna formuleras som *Flaggan för Auktoriserad ska alltid ha värde Sant, för att ett nytt möte ska kunna skapas.* Huruvida detta mål uppfylls eller inte beror endast på planeringsverktyget. Målet är därmed begränsat till en enskild agent och befinner sig på den lägsta abstraktionsnivån. Inom KAOS innebär det att målet kan anses vara ett *krav*. Ett krav definieras som ett mål kontrollerat av en enda agent.

Ett mål som berör flera agenter kan även det vara ett krav, då det viktiga är att det *kontrolleras* av en agent, exempelvis: *När inbjudningslistan är godkänd av*

mötesskaparen ska planerad tid och plats för mötet skickas till deltagarnas på inbjudningslistan e-postadresser.

Målet ovan berör flera agenter, men planeringsverktyget har ensam kontroll över att uppfylla målet. För att skicka e-post krävs dock en infrastruktur vilket inte nämns i målet ovan. Det går dock definiera en *förväntan* om systemets kontext som lyder *Inbjudningar som skickas till e-postadresser levereras till den adresserade personens e-postlåda*. Förväntningar består av mål som kontrolleras av enskilda agenter i systemets omgivning och är därmed något som systemet själv inte kan kontrollera. I det här fallet innebär det i praktiken att infrastrukturen kring e-posthantering inte inkluderas som en del av det framtida systemet, utan får antas ha ett visst beteende. Ytterligare ett exempel på en förväntan är: *En inbjuden deltagare kommer att delta vid möten om meddelat datum och plats stämmer med de begränsningar personen uppgett.* Systemet har ingen möjlighet att säkerställa att deltagaren faktiskt kommer att närvara på mötet, och det är därmed inte ett krav utan en förväntning.

Det är viktigt att ta hänsyn till, och även beskriva, den kontext som systemet kommer att befinna sig i. Medan systemets mål består av så kallade preskriptiva utsagor består beskrivningen av systemets kontext av deskriptiva utsagor. De deskriptiva utsagorna, som visas i Tabell 4, kan klassificeras som antingen *domänegenskaper* eller *domänhypoteser* beroende på hur stabil deras sanningshalt är. En deskriptiv utsaga som *Ett möte involverar minst två deltagare* är något som alltid kommer gälla och benämns då som domänegenskap. Domänhypoteser däremot är utsagor som är sanna under vissa förutsättningar eller är något som kan förändras som till exempel *Lördagar och söndagar är vid mötesplanering exkluderade dagar*.

Tabell 4 Typer av deskriptiva utsagor

Term	Beskrivning
Domänegenskap	Utsagor om systemets kontext vars tillstånd alltid gäller
Domänhypotes	Utsagor om systemets kontext vars tillstånd kan variera

3.5.1.2 Klassificering av mål

Då begreppet mål är det mest centrala inom KAOS finns ett utförligt klassificeringssystem där van Lamsweerde (2009) delar in mål i två olika dimensioner, *typ* och *kategori*. Det finns två typer samt två kategorier av mål. Uppdelningen mellan typ och kategori kompletterar varandra och är inte uteslutande. Det innebär att varje enskilt mål tillhör både en viss typ och en viss kategori.

Typer av mål:

- Behavioural goals beskriver systembeteende
- Soft goals beskriver preferenser mellan alternativa beteenden.

Kategorier av mål:

- Funktionella mål beskriver systemets funktion och syfte
- Icke-funktionella mål beskriver med vilken kvalité systemet ska fungera.

Typer av mål

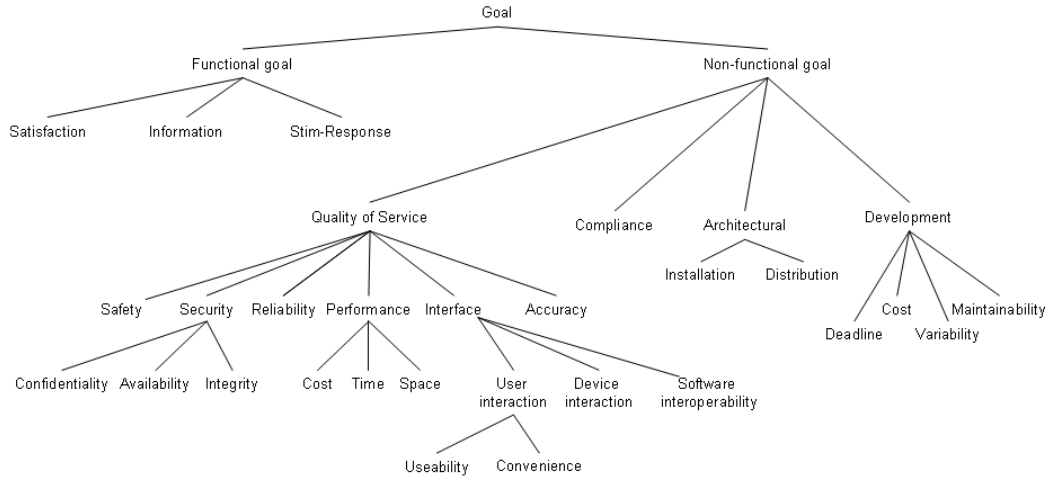
Behavioural goals beskriver vad systemet ska ha för beteende, och definierar indirekt en sekvens av tillståndsförändringar vilka styrs av en aktör. Ett exempel på behavioural goal är *Alla dörrar på tåget ska vara stängda när tåget är i rörelse*, och sekvensen är i det fallet (1) Tåget rör sig, dörrarna stängda, (2) Tåget har stannat, dörrarna är stängda, (3) Tåget står still, dörrarna är öppna, (4) Tåget står still, dörrarna är stängda, (5) Tåget rör sig, dörrarna är stängda. För att ett mål ska klassificeras som ett behavioural goal ska dess tillstånd alltid kunna avgöras tydligt och det ska alltid vara möjligt att avgöra om målet är uppnått eller inte. Behavioural goals kan delas upp i två undertyper, vilka båda kan vara i positiv eller negativ form: (1) *Achieve/Cease* och (2) *Maintain/Avoid*. Achieve goals är tänkta beteenden där ett tillstånd förr eller senare ska inträffa, förutsatt att ett annat villkor i systemet uppfyllts. Cease goal beskriver då beteenden som förr eller senare *inte* kommer att gälla. Maintain goals beskriver mål där önskade beteenden *alltid* ska gälla. Avoid goals kan användas för att uttrycka beteenden som *alltid* ska undvikas.

Soft goals är mål som beskriver ett beteende som föredras, men det går inte att tydligt säga om målet har uppfyllts eller inte. Ett exempel är *Passagerare ska ha tillgång till tydlig och översiktlig information om avgångar*. Soft goals kan däremot användas för att värdera olika lösningsalternativ genom att avgöra om ett visst lösningsalternativ uppfyller målet i högre grad än ett annat lösningsalternativ.

Kategorier av mål

Ett funktionellt mål beskriver en avsikt, som att planera smidiga möten. Ett icke-funktionellt mål föreskriver vilken kvalité systemet ska uppfylla, till exempel minimal interaktion mellan deltagare vid mötesplanering. Ibland likställs soft goals och icke-funktionella mål i litteraturen vilket inte är korrekt. Ett icke-funktionellt mål kan mycket väl vara klart definierat med en tydlig måluppfyllnad och därmed vara ett behavioural goal (van Lamsweerede, 2009; Werneck, Oliveira & Leite, 2009).

Kategorierna är inte heller uteslutande mot varandra och vissa mål kan därmed både klassificeras som funktionella och icke-funktionella, till exempel *Ambulansen ska vara på olycksplatsen inom 11 minuter* (som både beskriver funktion och kvalité). I Figur 1 visas de målkategorier som van Lamsweerde (2009) definierar.



Figur 1 Uppdelningen av målkategorier (van Lamsweerde, 2009, s. 269).

De funktionella målen kan delas in i;

- Satisfaction goals – att tillgodose aktörers begäran, exempelvis Varje begärt möte ska planeras om möjligt.
- Information goals – att hålla aktörer informerade om systemets tillstånd, exempelvis Alla inbjudna deltagare ska få information om mötet så snart som möjligt.
- Stimulus-Respons goals – att ge lämplig respons till specifika händelser, exempelvis Tidsplanering av möten ska ske så fort samtliga deltagare angett sina förslag.

De icke-funktionella målen kan klassificeras enligt en mängd underkategorier.

Klassificering av mål kan ge ett viktigt stöd vid kravhanteringen, bland annat genom att kontrollera att kravspecifikationen täcker alla relevanta kategorier eller identifiera konflikter.

3.5.2 I*

En mer agentorienterad ansats av GORE finns i form av ramverket i*. Utifrån i* har även ytterligare ansatser som TROPOS och GLR uppstått (Fricker & Glinz, 2010; Lapouchnian 2005). Till skillnad från KAOS, som fokuserar på systemets mål, ligger fokus i den agentorienterade ansatsen istället på systemets aktörer och systemets kontext. Dessa är grunden i systemmodellerna och varje aktör kan en rad egenskaper så som mål, förmågor, antaganden och åtaganden. Vid modelleringen beskrivs detta samtidigt som aktörens placeras i sin organisationskontext.

En aktör i de här ramverken kan antingen bestå av en agent, en roll eller en position. Agenter är konkreta aktörer så som datorsystem eller människor och varje agent har vissa specifika förmågor. En roll är däremot en abstrakt aktör med ett visst ansvar och vissa förväntningar gällande sitt handlande. En position består av en socialt igenkänd uppsättning roller vilka hålls av en agent.

Mellan aktörerna finns det olika former av beroenden som beror av hur de försökte uppnå sina mål. En aktör kan antingen sträva efter att uppnå sina mål själv eller genom att förlita sig på andra aktörer. Förlitar de sig på andra aktörer kan det öka möjligheterna och kan till exempel uppfylla mål mer kostnadseffektivt. För att uppnå sina mål, klara av uppgifter eller anskaffa resurser kan aktörerna därmed göra sig beroende av andra aktörer. Genom sådana samarbeten kan varje aktör uppnå mer, samtidigt som ett beroende introducerar en risk. Aktörerna blir sårbara ifall den andra aktören inte kan uppfylla beroendet. Aktörer måste därmed vara strategiska och balansera mellan möjligheter och sårbarheter. Fyra olika typer av beroenden finns definierade: (1) Goal, (2) Softgoal, (3) Task och (4) Resource. I likhet med KAOS innebär Goal ett tydligt tillstånd som en aktör försöker uppnå, och softgoal ett mål där det inte går att avgöra om målet är uppfyllet eller inte. Task är en uppgift en aktör utför och Resource rör de resurserna aktörerna behöver. (Lapouchnian, 2005; Werneck, Oliveria & Leite, 2009)

Genom modeller som representerar aktörer kan de processer som sker inom organisationen åskådliggöras och modellerna ger en möjlighet att utforska orsakerna och motiven bakom systemen och dess processer. I de tidiga faserna av kravhanteringen stödjer dessa modeller förståelsen av varför ett nytt system behövs. I senare faser av kravhanteringen används istället modellerna till utvärdering av hur nya systemkonfigurationer och processer uppnår uppställda mål.

3.5.3 NFR framework

Lapouchnian (2005) beskriver NFR (Non Functional Requirement) Framework som en processorienterad ansats som fokuserar på att rationalisera utvecklingen utifrån de icke-funktionella kraven. Målet är att genom modellering av icke-

funktionella krav (NFR) identifiera positiva och negativa effekter av olika alternativ för dessa krav. Ramverket innehåller tre typer av softgoals:

- Softgoal motsvarar icke-funktionella krav
- Operationalizing softgoals är softgoals som inte går att bryta ner i fler submål. Motsvarar de tekniker som behövs för att uppnå högre softgoals.
- Claim softgoals möjliggör dokumentering av design rationale för till exempel uppdelning av mål, prioritering av mål etc.

3.5.4 Fördelar med GORE

De komplexa förutsättningar och kontexter som system ska implementeras i kan genom att formulera mål struktureras på ett naturligt och sammanhängande sätt. Genom att identifiera bakomliggande mål till specifika krav med varför-frågor och identifiera krav och alternativa lösningar genom hur-frågor byggs en sammanhängande struktur upp. GORE tvingar modelleraren att motivera och föreslå genomtänkt funktionalitet genom att beakta olika alternativ och kriterier för att välja alternativ. Varje krav måste kopplas till agenter och motiveras av övergripande mål. Ett mål som *Säker tågtransport* kan förfinas i exempelvis tre undermål: (1) *Undvik tågkollisioner,* (2) *Bibehåll hastighet under gällande hastighetsgräns*, (3) *Bibehåll stängda dörrar medan tåget rör sig.* Dessa kan sedan i sin tur brytas ner ytterligare. Genom att arbeta med att både bryta ner mål (top-down), samt att bygga samman mål uppåt (bottom-up) uppnås en spårbarhet från enskilda krav till övergripande affärsmål. Strukturen skapar spårbarhet genom att den visar motiveringen till samtliga krav, deras ”existensberättigande” dvs. kopplingen till målen, samt visar vilka aktörer som har intresse i varje krav. En sådan struktur kan förenkla kommunikationen mot intressenter som beställare och slutanvändare då de genom målen få en förståelse för systemet. Genom att resonera kring mål systemet ska uppnå är det även möjligt att hålla stora delar av beskrivningen lösningsoberoende. Målen på lägre abstraktionsnivåer går sedan att använda som utgångspunkt för att utforska ett flertal alternativa lösningar (van Lamsweerde, 2009).

Ett problem vid kravhantering är att avgöra när en komplett kravmängd nåtts. Genom att definiera systemets mål går detta avgöra eftersom kravmängden kan anses vara komplett om samtliga mål blir uppfyllda av kraven. Omvänt går det även att kontrollera om alla krav är nödvändiga, genom att kontrollera om det finns krav som inte är kopplade till något mål. Ett problem kvarstår och det är bedömningen om beskrivningen av mål är komplett. Det finns dock tekniker inom GORE för att öka komplettheten hos målmodellerna (van Lamsweerede, 2009).

Genom att ha formulerade mål på olika abstraktionsnivåer går det identifiera var det finns målkonflikter och vilka involverade aktörer det påverkar. Att hantera sådana konflikter på övergripande abstraktionsnivåer istället för på enskilda krav gör det möjligt att påverka de bakomliggande faktorerna och konflikthanteringen blir på så sätt mer effektiv (van Lamsweerede, 2009).

3.5.5 Reflektion

GORE fokuserar på en rad begrepp som mål, förmågor och enskilda aktörer på ett sätt som ej görs i traditionell kravhantering. En intressant koppling mellan GORE och Försvarsmakten är användandet av begreppet förmåga. En fråga att lösa är hur dessa kan kopplas samman och hur detta kan hänga ihop med MODAF. Ytterligare en aspekt av GORE som är av intressant är hur det kan kopplas till tidigare forskning kring behov (Hallberg, Pilemalm, Westerdahl et al, 2008).

3.6 Kvalitetssäkring av krav

Att basera systemutvecklingen på krav av hög kvalité ökar sannolikheten avsevärt att erhålla ett lyckat resultat. Krav av hög kvalité innebär att det är rätt krav, men också att kraven är formulerade korrekt. Det senare är något som inte rönt lika omfattande uppmärksamhet. Vid FOI har en metod utvecklats som beaktar hur krav formuleras, granskas samt kvalitetssäkras avseende dess form, baserat på en litteraturstudie (Hansson, Granlund, Hallberg, Pilemalm & Pilemalm, 2010). Den utvecklade metoden består av nio attribut som krav ska formuleras och granskas utifrån, samt en process för att genomföra en strukturerad och formaliserad granskning. Detta avsnitt beskriver en utvärdering och förbättring av en metod för att kvalitetsgranska krav som utarbetades under 2010 i FoT-projektet *Arkitekturbaserad ledningssystemsutveckling*.

Det arbete som genomförts under 2011 inom *Kvalitetsbaserad ledningssystemsutveckling* har syftat till att utvärdera och förbättra den metod som togs fram under 2010. Utvärderingen har gjorts genom två separata granskningar av krav, där metoden har vidareutvecklats mellan granskningarna. De krav som granskats är krav hämtade ifrån Försvarsmaktens materielmålsättningar, även kallade Teknisk, Taktiskt och Ekonomisk Målsättning (TTEM). Resultatet av granskningarna har återförts till Försvarsmakten.

3.6.1 Metod för att granska krav avseende formulering

För att systematiskt granska krav efter den metod som beskrivs av Hansson et al (2010) behövs en checklista och en handledning (Figur 2). Handledningen utgör en förklaring till de attribut som förekommer i checklistan. Avsikten med attributen i checklistan är att skapa kriterier som krav ska uppfylla för att ses som välformulerat. Om de uppsatta attributen inte uppfylls vid en granskning bör en omformulering av kravet genomföras. Handledningen är även tänkt att utgöra ett stöd vid formulering av krav. Aktiviteterna att formulera krav och granska krav bör utföras efter samma förutsättningar, det vill säga nyttja samma handledning.

Krav	Form	Osammansatt	Specificerat	Entydigt	Verifierbart	Terminologi	Lösningsoberoende	Spårbart	Kommentar
Systemet skall hantera geodata i olika geodetiska referenssystem: RT90, SWEREF 99, SWEREF 99 TM, geografiskt WGS 84, UTM baserat på WGS 84.	X	-	-	X	-	-	X	X	Kravet är sammansatt, ej verifierbart och innehåller icke definierade termer
Då svenska förbands namn och anropssignaler kan vara svenska, ställer detta krav på att systemet kan hantera ÅÄÖ.	-	X	-	X	X	X	X	X	Kravet följer ej mall och är ospecificerat
Systemet skall ge stöd för hantering och uppdatering av grunddatabibliotek. Med detta avses stöd för tillverkning och uppdatering av grunddata, hantering av identiteter så att dessa blir lika i alla installationer, stöd för att skapa nödvändiga referenser mellan grunddata av olika slag samt stöd för konsistenskontroll.									
Systemet skall innehålla stöd för mottagning av fordonsuppföljning via NFFI.									

Figur 2 Checklista för granskning, där ett X betyder att kravet uppfyller attributet och – betyder att attributet inte uppfylls i kravet (Hansson, Granlund & Hallberg, 2011).

Den procedur som används för att granska krav avseende formulering benämns inspektion och fordrar fem aktiviteter; (1) översiktsmöte, (2) individuell granskning, (3) inspektionsmöte, (4) korrigering och (5) uppföljning (Hansson et al, 2010). Vid *översiktsmötet* deltar de personer som ska medverka i granskningen för att introduceras till kravmängden och till den tidsplanering som gäller för granskningen som helhet, samt för att erhålla nödvändig utbildning för uppdraget. Granskarna bör vara minst två, beroende på projektets storlek.

Under den *individuella granskningen* genomförs granskningen på ett krav i taget, där varje granskare ansvarar för sin tilldelade kravmängd. Granskningen genomförs baserat på checklistan och varje krav bedöms utefter samtliga kriterier. Under *inspektionsmötet*, där samtliga granskare deltar, samlas synpunkter avseende formuleringar och icke enhetliga krav in. Dessa diskuteras och sammanställs därefter till en lista på defekter. Defekterna *korrigeras* av en ansvarig granskare. Dessa tre steg, individuell granskning, inspektionsmöte och korrigering, upprepas tills dess att dokumentet i sin helhet är granskat och korrigerat.

Den avslutande *uppföljningen* äger rum i syfte att bedöma granskningens utförande som helhet. Uppföljningen sker med en representant för granskarna och med en representant för de som ursprungliga önskat erhålla granskningen samt med någon som har mandat att godkänna/underkänna kravdokumentet.

3.6.2 Utförande av kravgranskningar

För att utvärdera och förbättra den befintliga metoden för kravgranskning genomfördes två granskningar. Den första granskningen genomfördes på en databas med sammanställda krav hämtade ifrån materielmålsättningar. Den andra granskningen genomfördes på ett utkast till materielmålsättning. De genomförda granskningarna följde de ovan beskrivna aktiviteterna där aktiviteten *uppföljningen* fungerade som delgivning av resultatet till Försvarsmakten, som medgav möjlighet till återkoppling. Till stöd för båda granskningen användes checklistan samt handledning, dock uppdaterades checklistan och handledningen inför den andra granskningen utifrån erfarenheterna som erhölls vid den första granskningen. Under den individuella granskningen testades två olika förfaranden att genomföra kravgranskning. Det första förfarandet som användes var att granskning av enskilda krav särskildes från omformuleringen av krav medan i det andra förfarandet integrerades granskningen med omformuleringen av kraven. Omformuleringen syftade till att ge förslag på hur granskade krav skulle kunna formuleras för att åstadkomma att fler av de uppsatta kriterierna uppfylldes. De granskade kraven sammanfogades i fallet med kravdatabasen till ett granskningsdokument med granskningskommentarer och förslag till omformuleringar. I fallet med utkast till materielmålsättningen sattes ett förslag ihop till ett nytt utkast.

Genomförandet av granskningarna var koncentrerade till två-tre dagar under våren 2011. Sammanlagt genomfördes granskningarna av fyra personer, samtliga anställda vid FOI, där två av granskarna varken kände till den valda metoden eller hade arbetet med kravgranskning innan arbetet. Avsikten var att pröva om metoden är användbar även för en oerfaren granskare och att hitta förbättringsområden som är svåra att identifiera för de personer som redan använt metoden och vant sig vid arbetssättet. Parallellt med det individuella arbetet genomfördes kontinuerliga diskussioner för att ge granskarna en sammanhållen syn på tolkningar av olika krav och begrepp.

3.6.2.1 Granskning kravdatabas

Den första granskningen av krav genomfördes på den databasen innehållande sammanställda krav. Granskningen genomfördes med utgångspunkt i den befintliga checklistan och handledningen beskriven i Hansson et al (2010). Förutom de uppsatta attributen innehöll checklistan även en kolumn för kommentarer. Detta för att ge möjlighet att förklara bedömningen av kravet. För att bedöma kravet adderades ett fält till checklistan för att bedöma och ge varje granskat krav en status.

De nio attribut som den första granskningen genomfördes emot var i för granskningen gällande ordning;

- Specificerat - kravet ska bara kunna tolkas på ett sätt, innehålla tillräckligt med information men ingen onödig information samt innehålla vem/vad som skall utföra/uppnå något
- Lösningsoberoende - kravet ska inte uttrycka design eller på annat sätt påverka implementeringen.
- Förståeligt - kravet ska vara förståeligt, det vill säga ska kravet förstås utan ytterligare förklaringar.
- Tvetydigt ordval - ord som har en inbyggd mångtydighet ska inte användas, till exempel snabb, god eller robust.
- Osammansatt - krav ska kunna stå ensamt och endast uttrycka ett krav. Det vill säga kravet ska ej vara sammansatt av flera krav utryckta med avskiljare så som kommatecken, och eller samt. Krav ska inte heller vara uttryckt i punktform.
- Verifierbart - krav ska ha ett mätbart numeriskt värde samt enhet och/eller vara kontrollerbart som sant eller falskt.
- Spårbart - krav ska vara spårbart med ett unikt Id.
- Mall - för att krav ska följa mallen ska kravet inledas med ”Systemet skall …”
- Terminologi - för att krav ska följa projektets terminologi ska samtliga de begrepp och förkortningar som används i kravet finnas definierade i *TTEM*.

De bedömningar av krav som gjort gavs ett utav följande status:

- OK - kravet bedöms som rimligt.
- Omformulering - kravet bedöms kunna omformuleras till ett rimligt krav utifrån befintlig information.
- Mer info - mer information behövs för att kunna tolka och omformulera kravet.

Krav i denna granskningen var numrerade med unikt Id vilket gjorde att attributet spårbart inte behövde bedömas för varje krav utan samtliga krav var per automatik spårbara.

3.6.2.2 Granskning utkast till materielmålsättning

Den andra granskningen genomfördes på utkast till materielmålsättning. Granskningen genomfördes enbart av de krav som var numrerade i utkastet. Dokumentet tycks även innehålla krav skrivna i avsnitt i form av löptext, vilket gjorde att dessa inte hade tilldelats ett unikt kravId och varpå dessa inte har granskats.

Efter den första granskningen gjordes förändringar i checklistan och handledningen. Till den andra granskningen togs attributet *förståeligt* bort och ansågs ingå i *specificerat*. Motivet till detta är att krav som inte är förståeligt inte heller är specificerat då specificerat säger att krav inte ska vara tolkningsbara på flera sätt samt att krav ska gå att använda (för att implementera). Om krav inte är *förståeligt* uppnås inte dessa kriterier och kan därmed detta anses ingå i *specificerat*. Ordningen av attributen förändrades för att lättare passa hur granskningen gjordes. Även namnen på attributen förändrades för att lättare förstås i checklistan. Den andra granskningen genomfördes baserat på åtta attribut, i gällande ordning; (1) form, (2) osammansatt, (3) specificerat, (4) entydigt, (5) verifierbart, (6) terminologi, (7) lösningsoberoende och (8) spårbart. Även i denna granskning var samtliga krav från början kravnumrerade med unikt Id vilket gjorde att detta attribut inte granskades för varje enskilt krav.

Till den andra granskningen lades fyra nya kommentarsfält till i checklistan. Dessa var *definiera*, *metod*, *tvetydigt ord* samt *nytt ord*. Anledningen till utökningen var främst avsedda att ge stöd till granskarna under granskningen samt i kommunikation till uppgiftsgivaren om hur det resonerats runt kravet. Dessa ska dock inte ses som en utveckling av metoden utan som ett anpassat stöd av checklistan för denna granskning. Kommentarsfälten utgjordes nu av:

- Kommentar - fältet användes i de fall granskaren behövde tydliggöra något vid granskningen av kravet.
- Status med nivåerna:
    - OK:
    - Omformulera: Krav kan omformuleras utifrån tillgänglig information
    - Mer information: Mer sakinformation/domänkunskap behövs för tolkning.
    - Definiera: Kravet går att omformulera, men ett begrepp, term eller förkortning behöver definieras.
    - Metod: Granskningsgruppen behöver diskutera tolkningen av granskningsprotokollet.
- Tvetydigt ord - de olika tvetydiga ord som användes i kravet plockades ut med syfte att sammanställas i en lista.
- Nytt ord - det ord granskaren använde för att begränsa mängden tvetydiga ord noterades.

3.6.3 Utfallet av genomförda granskningar

Vid den första granskningen granskades 405 krav och i den andra 126 krav. I den första granskningen undgick endast 35 krav anmärkningar vilket utgör 8,6 % av kraven. Vid den andra granskningen undgick endast 3 krav anmärkningar vilket utgör 2,4 %. I Tabell 5 redovisas andel av kraven som fick anmärkningar indelat utifrån de olika attributen, (*spårbart* finns inte redovisat i tabellen då de inte granskats i och med att alla krav varit spårbara från början).

Tabell 5. Andel anmärkningar på krav för kravdatabasen samt utkast till materielmålsättning för respektive attribut i procent.

Orsak till anmärkning	Kravdatabas	Utkast materielmålsättning
Följer ej formen	35,5 %	76,2 %
Sammansatt	46,2 %	54,0 %
Ej specificerat	58,5 %	48,0 %
Tvetydigt ordval	51,6 %	34,0 %
Ej verifierbart	57,3 %	43,4 %
Odefinierad terminologi	27,7 %	24,5 %
Lösningsberoende	11,4 %	16,0 %

Under de första skedena av granskning av kravdatabasen utfördes enbart granskning av krav, då den metod beskriven i Hansson et al. (2010) som granskningen bygger på förespråkar att formulering och omformulering utförs av andra än de som utför granskning av form. Under detta första skede genomfördes granskningen särskilt från formuleringen och resulterade i en tid på 5 min 30 sek per krav. Genom att förändra granskningsförfarandet till att omformulera kravet samtidigt som granskningen genomfördes upplevde granskarna en tidsvinst även då tiden per krav ökade. Tiderna divergerade något mellan granskarna men ett resultat som iakttogs var att de längre tiderna spenderades av de granskare som hade mer erfarenhet av metoden än de som saknade erfarenhet.

Inför den andra granskningen omstrukturerades attributen i checklistan och det två attribut som vanligtvis granskades först under den första granskningen sattes först. Dessa två attribut var form och osammansatt. Metoden förändrades även för att effektivisera granskningen, vilket ledde till att granskningen av det kravet avbröts om den uppsatta formen inte följdes eller att kravet var sammansatt. Resultatet blev att omformuleringen av krav började tidigare och mer tid lades på att omformulera kravet. Detta upplevdes inte som negativt då vissa defekter är svåra att upptäcka då kravet till exempel består av mycket text som inte tillför kravet mer substans eller vid sammansatta krav. Resultatet blev att andra defekter upplevdes som enklare att upptäcka då kravet omformulerats och var något mer

överskådlig än den ursprungliga formuleringen. Efter varje omformulering gjordes en ny granskning av det omformulerade kravet för att säkerställa att samtliga attribut beaktats när det omformulerats. Mängden krav i den andra granskningen förändrades då de sammansatta kraven delades upp. Den totala kravmängden blev 210 krav efter de sammansatta kraven brutits upp, till skillnad från den ursprungliga mängden på 126 krav.

3.6.3.1 Reflektioner

Resultatet visar att utifrån de kriterier granskningen utfördes var en alltför liten del krav helt godkända avseende form. Utfallet tyder på att de kriterier som användes vid granskningen inte var de kriterier som var givna vid formuleringen. Det är till fördel för samtliga moment av kravhanteringen att de kriterier för granskning av form är kända för dem som är involverade i att formulera krav.

De attribut som använts vid dessa granskningar har arbetats fram ur en litteraturstudie och bygger på kända defekter. Detta kan ses som en brist då bara kända brister omhändertas och inga nya defekter identifieras. Detta är dock ett problem som alla checklistor lider utav.

Under granskningen av kraven har omformuleringar gjorts i möjliga fall. Vid ett normalförfarande ska omformuleringen göras av den ursprungliga författaren och inte granskaren (Fagan, 1986). Erfarenheter från de två granskningarna som beskrivs visar att i vissa fall kan omformulering och granskning kombineras. Det vinns tid och upplevs mer tillfredsställande av granskarna. Viktigt är dock att de omformulerade kraven återkopplas till författaren som gjort den ursprungliga formuleringen för att säkerställa att kravets innehåll inte förändrats.

Ett av målen med den metod som utarbetats var att granskningen ska kunna ske med avsaknad av domänkunskap. Detta har i vissa fall visat sig vara svårt då granskarnas avsaknad av domänkunskap i då vissa attribut granskats. Det största hindret har varit vid granskning av *förståeligt* och *lösningsoberoende*.

I vissa fall uttrycktes krav som sammansatta, till exempel i en lista. I de fallen saknas domänkunskap i det att granskarna inte kan avgöra om listan verkligen är komplett och sålunda borde uttryckas som enskilda krav, eller om listan bara utgjorde en exempellista, och sålunda underförstått borde kompletteras av den som bryter ner kraven till en teknisk specifikation. För tydlighet gäller att sammansatta krav aldrig ska användas.

En viktig iakttagelse under granskningarna var att terminologi listan inte enbart ska användas för att förklara akronymer utan även för att klargöra vilka tvetydiga ord som projektet accepterar och definiera dessas innebörd. Tvetydiga ord kan inte alltid undvikas men användandet kan begränsas och likriktas.

Under granskningen av utkast till materielmålsättning granskades enbart krav som funnits uttryckta som enskilda krav och försedda med ett kravId. Krav ska inte skrivas som löptext då dessa är svåra att identifiera och lätt kan förbises samt

för att underlätta spårbarhet ska krav vara försedda med en unik identifierare. Ytterligare en reflektion under granskningen var att flera krav var skrivna på en detaljerad nivå och tenderade till att uttrycka lösningar. Detta är något som avråds då lösningar inte tillhör kravhanteringen utan utformningen av designen. Att uttrycka lösningar i krav kan ge omotiverade och kostnadsdrivande begränsningar till möjliga realiseringar.

Att följ en strukturerad metod med genomarbetade kriterier redan vid formulering av krav är eftersträvansvärt. Detta för erhålla en formmässigt god kvalité på krav för att fokusera resurser till utarbetning av innehåll i krav.

3.7 The Physics of Notations

Detta kapitel beskriver Moodys teori för hur man skapar kognitivt effektiva modeller, en teori som Moody kallar ”The physics of notations” och vars kärna är nio principer. Följande avsnitt ger en beskrivning av principerna och i slutet av kapitlet ges reflektioner kring principerna och deras tillämpning.

Enligt Moody består en visuell *notation* (även kallat visuellt språk eller grafisk notation) av:

- En uppsättning grafiska symboler (visuell vokabulär) som linjer, ytor, volymer, etiketter (eng labels) och spatiala relationer.
- En uppsättning kompositionella regler (visuell grammatik)
- Definitioner som anger innebörden av varje symbol (visuell semantik)

Ett giltigt uttryckt i en visuell notation kallas för visuell mening, eller *diagram*. Ett diagram består av symboler, arrangerade enligt reglerna i den visuella grammatiken.

Grunden i Moodys teori är att en bra notation är *kognitivt effektiv*, vilket innebär att den snabbt och korrekt kan bearbetas och tolkas i av den mänskliga hjärnan. Att ha kognitiv effektivitet som designmål ger en operationell och mätbar definition av vad som är ”en bra notation”, till skillnad från designmål av mer subjektiv karaktär, som ”enkelt”, ”tilltalande” eller ”uttrycksfullt”. Att omvandla information till en bild innebär inte att den per automatik säger mer än tusen ord. Det måste finnas en förklaring till designvalet (eng. design rationale) – det räcker inte att förlita sig endast på sunt förnuft.

Teorin har en beskrivande del där Moody utifrån en stor mängd befintlig forskning kring kommunikation, grafisk design, visuell perception, kognition och semantik förklarar och förutser varför vissa notationer är mer effektiva än andra. Utöver den beskrivande delen av teorin har Moody även en föreskrivande del där han har formulerat nio principer för att designa kognitivt effektiva notationer.

Teorin har tagits fram för tillämpning på notationsnivån, ej diagramnivån. I diagramnivån ligger estetiska frågor såsom att symbolerna har samma skala/storlek, ligger i nivå och har lämpligt typsnitt.

Teorin omfattar inte heller semantiken, det vill säga att avgöra vilka begrepp och metamodellkonstruktioner som ska finnas i modellen samt vad de betyder – alltså vad som ska kunna visas med notationen.

3.7.1 Moody's principer

Moody's nio principers innefattar:

- Semiotic clarity - det bör vara ett 1:1-förhållande mellan semantiska begrepp och grafiska symboler.
- Perceptual discriminability - symboler ska vara tydligt åtskiljbara från varandra.
- Semantic transparency – symbolers utseende ska tydligt ange deras innebörd.
- Complexity management - ett diagram ska inte innehålla för många symboler.
- Cognitive integration - när det finns många diagram i en modell behöver det finnas lösningar för att förstå helheten, hur de hänger ihop och hur navigering ska ske mellan diagrammen.
- Visual expressiveness - utnyttja hela spektrat och möjligheterna för visuella variabler.
- Dual coding - använd text för att komplettera grafiken.
- Graphic economy - håll det totala antalet symboler i en notation kognitivt hanterbar.
- Cognitive fit - använd olika visuella dialekter för olika uppgifter och målgrupper.

3.7.1.1 Semiotic Clarity

Principen Semiotic clarity (sve. semiotisk tydlighet) innebär att det ska vara en 1:1 mappning mellan semantiska konstruktioner och grafiska symboler (Moody, 2011a). Det betyder att varje symbol oberoende av kontext endast ska betyda en sak, samt att alla begrepp inom notationen har en symbol.

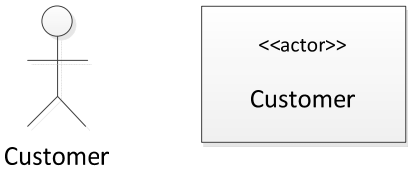
Semantiska konstruktioner är idéer eller begrepp, som kan förmedlas via grafiska symboler till andra personer. De semantiska konstruktionerna definieras i en notations metamodell (Moody, 2011b.). Den grafiska symbolen blir därmed en representation av det bakomliggande begreppet. Att det ska vara en 1:1 mappning betyder att det inte ska finnas flera olika grafiska symboler som betecknar samma begrepp (liknelse: synonym/synograf), och det ska inte finnas flera begrepp som representeras av likadana symboler (liknelse: homonym/homograf). Det ska heller inte finnas grafiska symboler som saknar bakomliggande begrepp eller vice versa. Det fyra fenomenen som kan uppkomma om inte principen följs är enligt Moody:

- Ett begrepp ingen symbol: Symbolbrist
- Ett begrepp med flera symboler: Symbolredundans
- Flera begrepp med samma symbol: Symbolöveranvändning
- Inget begrepp men en symbol (alt. Visuellt brus): Symbolöverflöd

Utifrån dessa begrepp kan därmed en notation analyseras utifrån semantisk och grafisk komplexitet. Moody (2011b) utvärderade notationen BPMN och visade att notationen innehåller 98 olika semantiska konstruktioner (begrepp) vilket blir notationens semantiska konstruktion. Notationen innehåller dock 177 grafiska symboler, vilket i sin tur definierar språkets grafiska komplexitet. Utifrån detta beräknas ett symbolbalansnetto på +79 vilket vidare kan beskrivas utifrån de fyra kategorierna av fel där Moody hittade 7 fall av symbolredudans, 5 fall av symbolöveranvändning, 2 fall av symbolöverflöd och 29 fall av symbolbrist.

Symbolredundans

Symbolredundans kan liknas med synonymer inom vanligt språk, till exempel bil och automobil. Hos Unified Modeling Language (UML) exemplifierar Moody (2011a) det med symbolerna för aktör (eng. actor), där symbolen för detta kan vara en ritad streckfigur eller en rektangel vari det står ”<actor>” (Figur 3). I exemplet nedan representeras aktören av kund (eng. customer).



Figur 3 Grafiska symboler i UML som båda betecknar begreppet aktör (Moody, 2011a, s.21)

Symbolredundans innebär att användarna av notationen måste välja vilken symbol som ska användas, och de som läser notation tvingas komma ihåg flera representationer för samma begrepp (Moody, 2009).

Symbolöveranvändning

Vid överanvändning av symboler innebär det att samma symbol används för att beteckna olika begrepp. I UML kan detta ses i att samma symbol (en tabell) används för klass, objekt och egenskaper (Figur 4).

Figur 4 Samma symbol i UML används för flera begrepp (Moody, 2011a, s.22)

Enligt Moody (2009) innebär symbolöveranvändning det värsta brottet mot Semiotic clarity. Genom att ha samma symbol för flera olika begrepp ökas risken för tvetydigheter och risken för feltolkningar. Kopplingen mellan en symbol och dess begrepp kan i bästa fall göras utifrån den kontexten som symbolen uppträder i. Detta bryter dock mot en fundamental regel i principen gällande tvetydighet, det vill säga att alla symboler ska ha en specifik betydelse, definierad i förväg och oberoende av kontext.

Hela symboler behöver inte återanvändas för att det ska bli problem. Symbolerna i BPMN är uppbyggda av delar som återanvändas inom flera olika symboler. Detta leder till delvis överlappande symboler med olika betydelser. Misstag kan då ske genom att läsare överför en betydelse från den ena symbolen till andra.

Symbolöverflöd

När en symbol används och inte syftar mot ett bakomliggande begrepp kallas det symbolöverflöd. Ett exempel på detta är kommentarer i UML (Moody & van Hillegersberg, 2009). För att klargöra detaljer i UML-diagram kan textrutor med förklarande text läggas in. Moody anser att detta endast ökar diagrammets komplexitet och risken att textrutan tolkas som om den stod för något mer. Moody föreslår att kommentarer endast skrivs i text, och inte text i en symbol.

Symbolbrist

Om ett begrepp existerar i notationen, men inte har någon symbol kallas detta symbolbrist. Moody (2009) verkar dock inte se detta som en allvarlig brist, utan att det snarare är värre om samtliga begrepp representerades i diagram.

En viss nivå av symbolbrist är därmed eftersträvningsvärt i notationsspråk för att hålla nere den grafiska komplexiteten. Moody (2011b) föreslår till och med att symbolbristen borde ökas i BPMN för att reducera den grafiska komplexiteten.

3.7.1.2 Perceptual Discriminability

Perceptual discriminability (motsvarande sve. Framträdande särskiljning) handlar om hur lätt symboler särskiljs från varandra visuellt. Särskiljning är en förutsättning för att modeller ska tolkas korrekt.

Visuell distans

Hur väl symboler skiljer sig åt bestäms av den visuella distansen (eng. visual distance) som existerar mellan symbolerna. Med detta avses med hur många variabler (till exempel färg, form eller storlek) symbolerna skiljer sig åt visuellt, samt storleken på skillnaderna mellan symbolerna. Ju mer symbolerna skiljer sig åt desto snabbare och mer exakt tolkas symbolerna. Många notationer har låg särskiljning då de använder sig av former och sammankopplande linjer som är relativt lika. Enligt Moody (2009) visar studier att entiteter och relationer ofta förväxlas i ER-diagram (Figur 5).

Figur 5 Exempel på entiteter och relationer i ER-diagram, vilket ofta är en orsak till missförstånd (Moody, 2009, s. 763)

Perceptuel Popout

Genom att ha skillnader på minst en unik visuell variabel mellan symboler uppnås det som kallas Perceptuel popout. Med detta menas att symboler framträder på ett tydligare sätt i modeller, vilket leder till effektivare särskiljning. Symboler som har en liten skillnad eller särskiljer sig med en kombination av befintliga variabler kräver mer inspektion för att förstås. Detta medför en större ansträngning för att tolka modellerna samt att risken för feltolkning är större. Dock är risken för feltolkning större hos personer som inte är vana att tyda modeller, jämfört med erfarna ”experter”.

Formens överlägsenhet

Enligt Moody (2009) visar studier att den variabel som effektivast särskiljer symboler är form. Ett sätt att förbättra ER-diagrammet skulle därigenom vara att ändra formen på relationen (Figur 6).

Figur 6 Exempel på hur entiteter och relationer i ER-diagram skulle kunna förändras till det bättre (Moody, 2009, s. 763)

En skillnad i form gör ofta att symbolerna tolkas som olika objekt. Användning av samma form, men olika särskiljare, kan vara ett effektivt sätt att skapa subtyper av ett objekt. Således kan symbolerna särskiljas, men ett visuellt intryck skapas att symbolerna härrör ifrån samma huvudtyp.

Redundant särskiljning

Redundant särskiljning innebär att använda flera visuella variabler för att förstärka särskiljningen, exempelvis färg (Figur 7). Att använda text som särskiljare för symboler är i ingen effektiv metod och förespråkas inte. Dock kan textuella begrepp bidra till kontextskapande eller användas för redundant kodning (eng. redundant coding).

Figur 7 Exempel på hur entiteter och relationer i ER-diagram skulle kunna förbättras med både förändrad form och färg (Moody, 2009, s. 763)

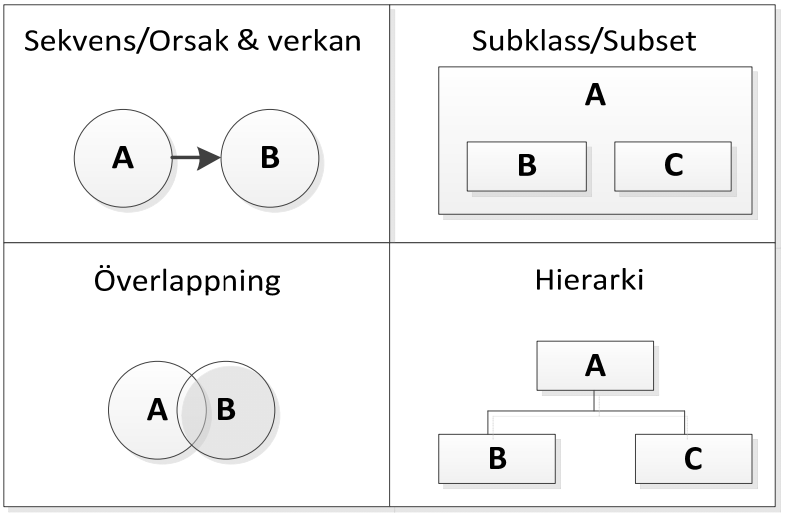
3.7.1.3 Semantic Transparency

Semantic transparency (motsvarande sve. Intuitiv semantik) handlar om hur väl innebörden av symboler kan härledas till deras utformning. Med detta menas att symboler som används ska representera deras innebörd, exempelvis kan användandet av en streckgubbe användas för att representera en människa (eller som i UML en aktör). Genom tillämpning av Semantic transparency minskas den kognitiva belastningen tack vare naturligheten, det intuitiva, i symbolerna. Om en symbol inte omedelbart kan förstås (tolkas) säger Semantic transparency att symbolerna ska vara lätta att lära sig. Detta medför att Semantic transparency inte kan mätas genom sant eller falskt utan har en mer flytande skala och delas in enligt följande:

- En symbol är omedelbart semantisk (eng. semantically immediate) om en novis kan tolka symbolens innebörd. Exempelvis när en person symboliseras med en streckgubbe.
- En symbol är semantiskt godtycklig (eng. semantically opaque) om det endast finns ett godtyckligt samband mellan symbolen och dess innebörd. Exempelvis när en person symboliseras med en rektangel.
- En symbol är semantiskt avvikande (eng. semantically perverse) om en novis troligen tolkar innebörden av symbolen på ett felaktigt sätt. Exempelvis när en person symboliseras med ett kuvert.

Semantiskt godtycklig kan ses som noll-läget på skalan där omedelbart semantisk är den positiva delen som eftersträvas, medan semantiskt avvikande är den negativa delen av skalan som bör undvikas. Mellan semantiskt godtycklig och omedelbart semantisk finns dock en varierande grad av självförklarande symboler, där det i vissa fall kan behövas en inledande förklaring för att symbolerna ska förstås. För att lyckas med att skapa intuitivitet bör symboler med en allmängiltig innebörd användas. Några exempel är symboler som har kulturella associationer som att rött betyder fara, metaforiska symboler som att korsade svärd betyder konflikt eller funktionsliknande symboler som att en papperskorg betyder borttagna objekt. Ett sätt att få noviser att tolka diagram korrekt är att använda bilder istället för abstrakta symboler.

Semantic transparency kan även användas för relationer. I diagram används ofta linjer mellan objekt för att påvisa en relation mellan dessa. Linjer kan betyda nästan vad som helst och ger ingen ledtråd om dess innebörd. Genom att använda andra tekniker, till exempel där symboler mer interagerar med varandra, kan förståelsen för relationen ökas (Figur 8).



Figur 8 Exempel på hur Semantic Transparency kan användas för relationer (Moody, 2009, s. 765)

3.7.1.4 Complexity management

Complexity management (motsvarande sve. Komplexitet i diagram) handlar om förmågan att presentera stora mängder information utan att överbelasta den mänskliga hjärnan. Detta avser antalet symboler eller tokens i varje diagram (inte

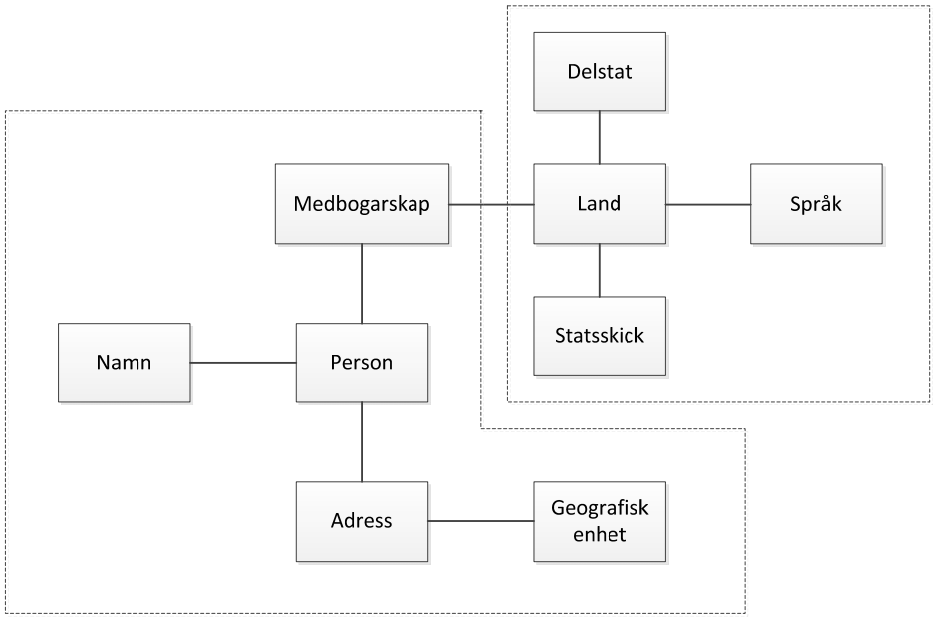
att förväxlas med principen Graphic economy som avser antalet symboler i notationen).

Den kognitiva effektiviteten påverkas i hög grad av komplexiteten eftersom mängden information som kan förmedlas via ett diagram är starkt begränsad av de mänskliga perceptuella förmågorna och de kognitiva förmågorna. Förmågan att särskilja element i ett diagram minskar när komplexiteten ökar. Dessutom begränsas förståelsen av arbetsminnets kapacitet (eng. working memory capacity). Generellt är den mängd information människan kan hantera i arbetsminnet 7 +/- 2 element, vilket innebär att även antalet element per diagram bör begränsas till detta antal. Komplexiteten i diagram är svårast att hantera för noviser som saknar strategier för att hantera komplexitet, och är en av de största barriärerna för att få noviser att förstå diagram.

Modularisering

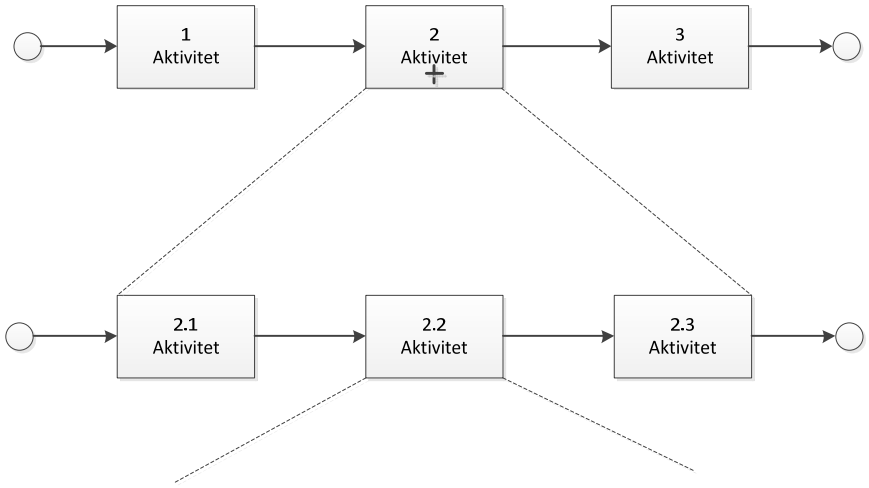
För att effektivt representera en komplex situation behöver notationer ge stöd för att dela upp diagram i perceptuellt och kognitivt hanterbara delmängder. Att dela upp helheten, ett system, i subsystem kallas för modularisering. Detta är ett effektivt sätta för att öka förståelsen där studier visar att modularisering kan öka förståelsen av diagram med 50% (Moody, 2009).

Modularisering kan uppnås på två olika sätt. Dels genom att komplexa diagram delas upp i flera diagram, utifrån lämpliga grupperingar ifrån det komplexa diagrammet. Dessa grupper representeras sedan i var sitt nytt diagram, där varje grupp utgör fokus för det nya diagrammet (Figur 9). Detta sätt bör kombineras med kontextualisering som beskrivs inom Cognitive integration.



Figur 9 Exempel på modularisering med fokus.

Det andra sättet är att skapa hierarkisk struktur (Figur 10), där det finns två alternativ, *top-down* och *bottom-up*. Båda alternativen resulterar i samma struktur, dock skiljer sig angreppssättet åt. Top-down strukturer skapas genom att från högsta nivån bryta ned elementen och skapa underdiagram på en lägre abstraktionsnivå. Bottom-up är motsatsen där strukturen byggs nedifrån och upp. Element på lägsta nivå grupperas ihop och diagram på en högre abstraktionsnivå skapas.



Figur 10 Exempel på modulariseirng med hierarkisk struktur.

Vilket sätt som väljs är beroende på situation då inte alla problem passar att lösas med en hierarkisk struktur och vice versa. Notationen bör ha stöd för hantering av komplexiteten, och då inte bara i form av symboler och tänkta konstruktioner utan även diagrammatiska konventioner för hur modulerna ska brytas ner. Dock är det en vanlig missuppfattning att detta är en verktygsfråga och inte en fråga för notationen. Om notationen inte ger stöd för dessa frågor riskerar det att leda till inkonsekvens i modellen och suboptimala lösningar.

Exempel på notationer som har detta stöd för modularisering är dataflödesdiagram och UML´s aktivitetsdiagram. Notationer som saknar detta stöd är exempelvis ER-diagram och i*.

3.7.1.5 Cognitive integration

Cognitive integration (sve. Kognitiv integration) är främst relevant då flera diagram används för att beskriva ett system, både flera diagram av samma typ (som en följd av modularisering i komplexitetshanteringen) och en svit av olika diagram (exempelvis UML-diagram och MODAF[5]). Att representera ett system i flera diagram ställer kognitiva krav på läsaren att mentalt integrera informationen från flera diagram och även hålla reda på var diagrammen befinner sig i modellen. För att multipla diagram ska vara kognitivt effektiva måste det finnas mekanismer för att stödja:

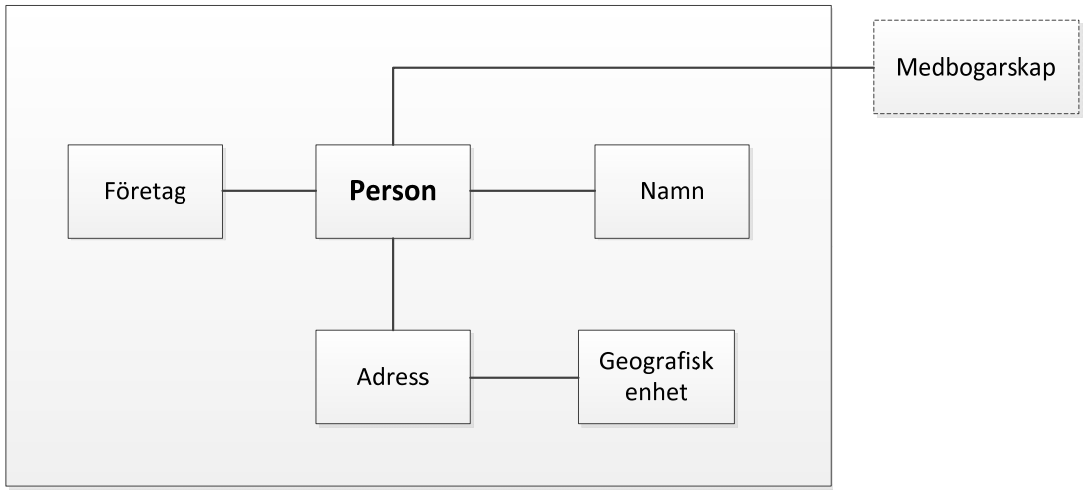
[5] http://www.mod.uk/DefenceInternet/AboutDefence/WhatWeDo/InformationManagement/MODAF/

- Konceptuell integration – mekanismer för att hjälpa läsaren att sätta samman information från olika diagram till en sammanhållen mental representation av systemet.
- Perceptuell integration – perceptuella ledtrådar för att göra navigationen mellan diagram enklare.

Konceptuell integration

En viktig mekanism för att lösa konceptuell integration är ett sammanfattande övergripande diagram, som presenterar en helhetsbild av systemet. I de fall en hierarkisk struktur tillämpas blir detta naturligt i diagrammet på den översta nivån, men i de fall olika typer av diagram tillämpas behöver en ny typ av diagram tas fram.

Kontextualisering (fokus+kontext) är en teknik som används där en del av systemet (fokus) visas i en kontext av systemet som helhet, alltså att visa de element det har relation till i andra diagram, som främmande element. Denna teknik stödjer både konceptuell integration genom att ge förståelse för varje element och dess relationer till andra element, samt perceptuell integration eftersom det underlättar navigeringen mellan diagram (Figur 11).



Figur 11 Exempel på kontextualisering.

Perceptuell integration

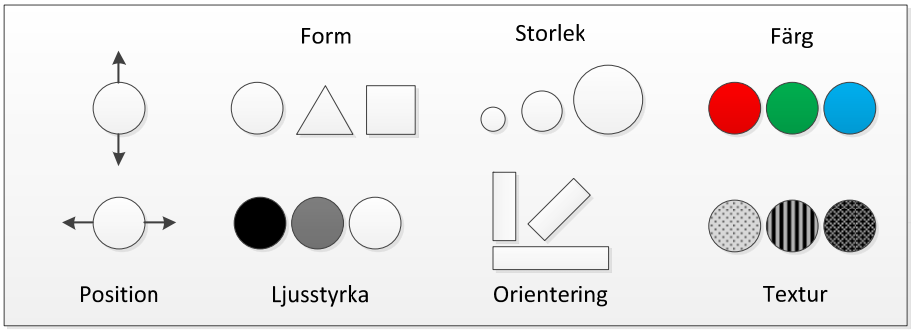
Perceptuell integration handlar om att hitta rätt väg, och innebär att följande frågor ska besvaras:

- Orientering – var är jag?
- Vägval – var kan jag gå?
- Route monitoring – är jag på rätt väg?
- Identifiering av målet – är jag framme än?

Exempel på tekniker för att underlätta perceptuell integration är tydliga etiketter på varje diagram (namngivning) och nivånumrering för att påvisa var i den hierarkiska strukturen man befinner sig. Att ha med navigationsledtrådar (signposting) stödjer vägvalet. En navigationskarta, innehållande alla diagram och navigeringsvägar mellan dem som stödjer orientering, route monitoring och vägval. Idag stödjer ingen existerande notation detta fullt ut (Moody, 2009).

3.7.1.6 Visual Expresiveness

Visual expresiveness (sve. Visuell uttrycksfullhet) handlar om hur många visuella variabler som används i en notation. De variabler som finns att tillgå när en modell eller notation utvecklas är färg, position, form, storlek, textur, riktning och ljusstyrka (Figur 12).



Figur 12 De åtta visuella variablerna som finns att tillgå vid skapandet av notationer (Moody, 2009, s. 761)

Den visuella uttrycksfullheten mäts genom att ange antalet visuella variabler som notationen använder (informationsbärande variabler) samt antalet visuella variabler som inte används (fria variabler). Många notationer använder bara en variabel, som exempelvis text eller form, och detta gör dem endimensionella. Kartor däremot innehåller många variabler (allt från färg och texturer till form och text) och de utnyttjar de åtta tillgängliga variablerna betydligt bättre än många notationer (Moody, 2009).

Färg är den variabel som anses mest kraftfull i relation till människans perceptiva förmåga – vi kan uppfatta tusentals olika nyanser och se skillnaden mellan dem. Färg är dock inte lämpligt att använda som enda urskiljningsfaktor i exempelvis

ett diagram då alla inte har samma färgseende. Istället bör färg kombineras med andra markörer som exempelvis form för att förstärka innehållet. Form är den faktor som är mest använd i visuell presentation, i synnerhet när det gäller att presentera information för dem som inte är insatta och invigda (noviser) i ämnet. Det finns oändligt med former men de är endimensionella och de former som främst används i notationer är rektanglar av olika slag och de hör inte till de former som är lättast att tolka (Moody, 2009).

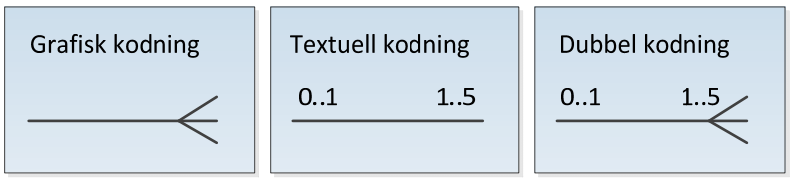
När man väljer visuell variabel är det viktigt att tänka på att formen som används ska följa innehållet i budskapet – egenskaperna i de visuella variablerna som används bör stämma överrens med egenskaperna i den information som presenteras. Till exempel kan färg och form användas för att representera nominella värden, medan storlek kan användas för intervall – färger och former har inte en speciell ordning till skillnad från storlek där det psykologiskt finns en ordning från minst till störst eller tvärtom.

För att maximera den visuella uttrycksfullheten ska grafisk kodning användas snarare än textuell. Genom att använda mer visuella och grafiska variabler än text så nyttjar man perceptuell bearbetning i hjärnan och minskar belastningen på övrigt tänkande.

3.7.1.7 Dual Coding

Enligt principerna om Perceptual discriminability och Visual expresiveness bör inte text användas för att koda information i visuella notationer. Dock finns det undantag – bilder och ord behöver inte utesluta varandra och enligt kognitiva teorier om dubbel kodning av sinnesintryck kan det vara bra att förmedla ett budskap via både text och bild. Principen Dual coding (sve. dubbelkodning) handlar om att förstärka eller förtydliga betydelsen av ett budskap genom att använda sig av fler än en kanal. Det kan till exempel handla om att använda både text och bild. Att använda text i en figur underlättar för noviser och kan minska risken för dubbeltydighet i figuren eller diagrammet. Det handlar dock inte om att ersätta bild eller figurer med text.

Dual coding kan göras genom att lägga in kommentarer i diagrammet (jämfört med att lägga den beskrivande texten i ett eget dokument som bilaga), eller förstärka en symbol genom att även textuellt beskriva den. I det senare fallet ger det mervärde till diagrammet då det ger möjlighet att ange det exakta värdet, och tydliggör innebörden av symbolen. Figur 13 visar hur Dual coding kan användas för att utöka informationen i en symbol genom att specificera minsta och högsta kardinalitet.



Figur 13 Exempel på hur dubbelkodning kan användas för att utöka informationsmängden i diagram (Moody, 2009, s. 771)

3.7.1.8 Graphic Economy

Graphic economy (sve. Grafisk sparsamhet) handlar om det totala antalet olika symboler (alt semantic constructs) i en notation, det vill säga storleken på dess visuella vokabulär. Graphic economy skiljer sig från Complexity management, som endast gäller antalet element i ett diagram. Användandet av en stor visuell vokabulär bidrar till komplexitet och försvårar förståelsen för diagram, speciellt för noviser. Om en symbol inte går att förstå rakt av måste en teckenförklaring (eng. legend) användas, vilket fördröjer läsningen av diagram. Ju fler symboler en notation har, desto svårare är det att uppfylla de övriga principerna.

Komplexiteten kan även ligga i antalet regler för användning av en viss symbol och informationsdensiteten i symbolen. Med detta avses exempelvis att om linjerna är prickade ger symbolen en viss innebörd, och om en extra symbol tillförs inuti symbolen får den en annan innebörd.

För att hantera grafisk komplexitet finns det fyra huvudsakliga åtgärder. Den första handlar om att reducera den semantiska komplexiteten genom att minska antalet begrepp som ska visas med symboler, kanske använda bara en symbol för olika typer av roller exempelvis.

Den andra åtgärden handlar om att dela upp den semantiska komplexiteten, det vill säga utöka notationens metamodell med fler diagramtyper, istället för att visa alla typer av information i en och samma diagramtyp. Detta minskar inte det totala antalet symboler i notationen, men begränsar antalet symboler till en hanterlig mängd som läsaren behöver ha kunskap om per diagramtyp.

Ytterligare ett sätt att hantera grafisk komplexitet är att introducera symbolbrist (eng. symbol deficit) – all information passar inte att visas i diagram. Man kan välja att inte visa information i grafisk form. Exempel på alternativa lösningar är textuell form i diagrammet, eller textuell form i ett tillhörande dokument, eller i en matris (exempelvis för relationer).

Det fjärde alternativet är att öka den visuella uttrycksfullheten i diagrammet. Begränsningen på att perceptuellt kunna särskilja på olika symboler är bara tillämpligt så länge endast en visuell variabel används (vilket de flesta notationer

gör – form). Om fler visuella variabler används ökar förmågan att särskilja mellan symboler och fler symboler kan användas.

3.7.1.9 Cognitive Fit

Inom systemutvecklingsområdet är det ett etablerat antagande att olika typer av representation av information är olika lämpade för olika uppgifter och målgrupper (Moody, 2009). Inom visuella notationer handlar Cognitive fit (avser ungefär sve. ändamålsanpassning) om att använda olika ”visuella dialekter” för olika uppgifter och målgrupper. Dessa dialekter ska då ses som kompletterande och inte konkurrerande.

Det finns två tungt vägande skäl till att skapa visuella dialekter för en notation:

- Olikheter mellan experter och noviser.
- Olika media för representation av diagram – rita för hand eller i ett datorbaserat verktyg.

Utöver dessa två faktorer finns ytterligare anledningar att använda sig av visuella dialekter, till exempel kulturella skillnader (när notationen används i olika kulturella kontexter) och individuella skillnader i verbal vs spatial förmåga (människor har olika förmåga att processa information i textuell och grafisk form).

Skillnaden mellan experter och noviser

De viktigaste skillnaderna mellan experter och noviser är i detta sammanhang:

- Noviser har svårare att skilja på olika symboler.
- Noviser måste medvetet komma ihåg vad symboler betyder.
- Noviser påverkas mer av komplexitet då de saknar strategier för att hantera informationen i delmängder/sjok.

Det kan vara lätt att tycka att alla diagram bör anpassas efter noviserna, då experterna även kommer att förstå dessa diagram. Men, ”the expertise reversal effect” motsäger detta - optimering av representationen för noviser kan reducera förståelsen för experter och vice versa. Detta innebär att det behövs minst två dialekter för notationer, en ”professionell” och en ”enklare”. Notationer som är designade för kommunikation med noviser behöver använda mer särskiljande symboler (eng. Perceptual discriminability), ha mindre komplexitet (Complexity management), följa konventioner för minnesstöd (Semantic transparency), ha förklarande text (Dual coding) och förenklad visuell vokabulär (Graphic economy).

Olika media för representation av diagram

Det största behovet av att anpassa notation efter medium gäller skisserna i de tidiga utvecklingsfaserna. Skisser görs ofta på blädderblock eller whiteboard-tavlor, för att i senare skede överföras till digitalt format i ett verktyg. Några av de största kraven som ställs på notationer som ska ritas för hand är:

- Perceptuell diskriminering – Det är svårare att vid handritning särskilja på snarlika symboler.
- Semantic transparency – bilder och ikoner är svårare att rita än geometriska former, speciellt för de som kanske inte besitter någon artistisk ådra.
- Visual expresiveness – en del variabler är svårare att utnyttja när man ritar för hand, så som färger och textur.

Behovet är därmed en förenklad visuell dialekt för skissande och en berikad notation för det slutliga diagrammet.

3.7.2 Reflektioner

I detta avsnitt resoneras runt principerna som Moody har skapat. Reflektionerna handlar främst om vad som saknas i Moodys resonemang och vad som behövs för att göra principerna tydligare. Förutom detta har även reflektioner gjorts angående hur principerna ska tillämpas.

3.7.2.1 Reflektioner kring de nio principerna

Moody tar upp ett antal värdefulla principer, och pekar i dem på problem, och i vissa fall på övergripande lösningar. Men för att kunna tillämpa principerna i en verklig situation vid skapande eller granskning av en modell ser vi att det saknas en del arbete för att principerna ska kunna ge något direkt stöd. Detta avsnitt beskriver några reflektioner kring Moodys principer, men ger främst beskrivning av vad som saknas i principerna för att de ska kunna ge ett konkret stöd.

Semiotic Clarity

Att ha symboler som har en förutbestämd betydelse tycks grundläggande, liksom att varje begrepp endast ska kunna representeras av en symbol. Anmärkningsvärt är att en notation som UML inte når upp till detta. Att undvika symbolöveranvändning, symbolredundans och symbolöverflöd bör vara självklart och går enkelt att kontrollera.

Det som främst kräver mer eftertanke och avvägning är hanteringen av symbolbrist. Symbolbrist är enligt Moody något eftersträvansvärt, men Moody påtalar inte hur en notation med symbolbrist ska utvärderas. En avvägning måste ha skett för att begrepp som inte tilldelas en symbol verkligen inte behöver en symbol. Hur en sådan avvägning görs och på vilka grunder besvaras inte av Moody.

Perceptual Discriminability

Inom Perceptual discriminability beskriver Moody ett antal principer för hur symboler effektivt kan särskiljas från varandra, vilket är önskvärt. För att uppnå denna särskiljning mellan symbolerna tillämpas medlen i principen för Visual expressiveness, med exempelvis färg och form.

Moody anger dock inte på vilket sätt de olika visuella variablerna kan tillämpas. Exempelvis menar Moody att den visuella distansen, det vill säga skillnaden mellan symboler, bör vara så stor som möjligt, men han ger inget tydligt exempel på hur den visuella distansen räknas ut. Moody hänvisar till studier som visar på att liggande och roterad fyrkant är för lika, men beskriver inte hur bedömningen att de är för lika har gjorts. Dessutom är det motsägelsefullt då ett sätt att använda sig av de visuella variablerna för att särskilja symboler är orientering och rotation, men i exemplet anser Moody alltså ändå att de är för lika. Det vore önskvärt att ha hjälpmedel för hur bedömningen av den visuella distansen ska göras. Risken med avsaknad av hjälpmedel blir att subjektiva bedömningar utförs.

För att uppnå Perceptuel popout anger Moody att symboler ska ha minst en unik visuell variabel, exempelvis storlek. Att ha symboler med olika storlek, till exempel en stor och liten cirkel innebär en uppenbar skillnad då symbolerna finns i samma diagram. Dock kan detta vara ett problem om symbolerna inte uppenbarar sig i samma diagram då det inte är uppenbart vilken som är den lilla respektive stora cirkeln. Därmed borde det vara så att vissa visuella variabler lämpar sig mer för att skapa framträdande särskiljning inom ett och samma diagram, medan andra variabler ger effekt även mellan olika diagram.

Semantic Transparency

Symboler som är semantisk avvikande (det visuella uttrycket har ingen koppling till den faktiska betydelsen) bör vara naturligt att undvika, däremot att ha symboler som är självförklarande kan vara både naturligt och samtidigt lite ansträngt. Vissa begrepp är mer naturliga och har mer eller mindre självförklarande symboler, så som streckgubbe för person. Andra begrepp är däremot svårare att hitta intuitiva symboler till och kan då bli mer ansträngt. För att avgöra vilken grad av omedelbart semantiska symboler som är lämpligt bör hänsyns tas till vilken målgrupp som diagrammet/modellen konstrueras för, där vissa kontexter kräver mer eller mindre intuitiv semantik. Till exempel kan användandet av intuitiva symboler i vissa sammanhang uppfattas som förringande och ställer då till med mer skada än nytta. Detta resonemang kring målgruppen återkommer i reflektionerna kring principen Cognitive fit.

Moody nämner kulturella associationer vid användandet av omedelbart semantiska symboler, och att detta hänsynstagande bör poängteras ytterligare. Det intuitiva i tolkningen av symboler kan vara subjektiv, och kopplat till en viss kultur (till exempel ett land eller en organisation). I vissa kontexter kan symboler

vara vedertagna och förstås men i andra kontexter är dessa symboler inte vedertagna och förstås inte. Ett närliggande exempel är kroppsspråk, där att nicka med huvudet kan betyda ”ja” i en kultur och ”nej” i en annan. På samma sätt kan en symbol ha en självklar betydelse i en kultur eller kontext men sakna denna eller ha en annan betydelse i en annan kontext.

Ett sätt att undvika missförstånd utifrån kultur eller kontext som gäller de flesta begrepp är att presentera dem med semantiskt godtyckliga symboler, vilka varken ger ledtrådar om dess innebörd, men inte heller missleder förståelsen för symbolen. Detta är dock inte något som Moody förespråkar.

Complexity Management

Att inte ha för många symboler i ett och samma diagram är grundläggande. Moody menar att det finns en gräns på sex symboler per diagram, vilket baseras på arbetsminnesstudier som visar att människan har förmåga att hantera 7 +/-2 element samtidigt. Dock anger Moody även (bland annat i principen Graphic economy) att om uttrycksfullheten ökar i diagrammet (Visual expressiveness) och symbolerna i diagrammet är mer åtskilda visuellt kan det maximala antalet symboler utökas, då hjärnan klarar av att uppfatta fler antal element inom samma tid. Det behövs därmed stöd för hur bedömningar och prioriteringar av komplexiteten och möjliga lösningsalternativ kan göras. Ta exemplet att ett diagram innehåller 12 symboler. Då är det kanske ändå enklare att visa alla symboler på ett och samma diagram, istället för att behöva läsa två diagram och därmed belasta läsaren med att behöva integrera diagrammen kognitivt och behöva läsa två diagram för att se helheten. Detta gäller speciellt då de aktuella symbolerna inte har någon naturlig skärningspunkt för uppdelningen. Det behövs mer konkreta riktlinjer för hur modularisering kan göras i olika typer av situationer och olika typer av diagram för att följa den här principen fullt ut.

Cognitive Integration

För Cognitive integration beskriver Moody ett antal riktlinjer som kan ge stöd för att underlätta förståelsen av modellens helhet och hur olika diagram hör samman, vilka bör kunna göra stor nytta. Moody ger dock bara kortfattade ledtrådar om lösningar vilket gör att det saknas en hel del information om konkreta sätt att skapa en kognitivt väl integrerad notation . Till exempel anger Moody att nagivationsledtrådar kan användas för att stödja vägvalet mellan diagram, men han anger inte vad en sådan navigationsledtråd kan vara och ger inte heller något exempel.

Visual Expressiveness

Principen om Visual expressiveness handlar om att utnyttja alla de olika grafiska uttryckssätt som finns tillgängliga. Denna princip skiljer sig därmed från de övriga då denna inte pekar på specifika problem, utan verktygen i denna princip är lösningen för flera övriga principer, exempelvis Perceptual discriminability.

Moody ger kartor som ett exempel på visuella notationer som nyttjar dessa uttryckssätt på ett bra sätt genom användandet av färger, former och texturer likväl som storlek och positioner. Kartor innehåller dock data som är lämpligt att representera på detta vis, då de är en förenklad representation av något som konkret – geografi. När det gäller att visuellt representera abstrakta fenomen, som exempelvis modeller av organisationer eller beslutsprocesser, finns det dock inte lika tydliga kopplingar till färg, form och position. Detta gör det svårare att utnyttja alla visuella variabler samtidigt som man hanterar övriga principer som intuitiv semantik och grafisk sparsamhet.

Sedermera anger Moody inget om hur de visuella variablerna ska användas. Vissa variabler, till exempel färg och storlek, kan skapa intryck av en innebörd i symbolerna som kanske inte är avsedd. Återigen exemplet med en stor och en liten cirkel, vilket kan tolkas att de härrör ifrån samma typ, där den stora cirkeln är överordnad den lilla, vilket leder till att information skapats som kanske inte är korrekt. Ett exempel, där Moody snarare förvirrar, är att han i principen Visual expressiveness anger att form är den minst kognitivt effektiva visuella variabeln, medan färg är en av de mest kognitivt effektiva sett till människans förmåga att särskilja mellan många olika varianser i färger. Men i principen för Perceptual discriminability anger han att form är den variabel som är bäst lämpad att använda för att särskilja mellan symboler, utan att förklara hur detta påstående förhåller sig till hans tidigare.

Dual Coding

Principen om Dual coding handlar om att öka läsbarheten eller förståelsen i ett diagram eller en modell genom att använda sig av både text och bild, genom kommentarsfält, etiketter eller text som förstärker eller förklarar symboler.

Att använda sig av Dual coding kan vara ett sätt att ta sig runt andra brister i en notation, till exempel om symbolerna inte är intuitiva eller konsekventa. Text som förstärkning av en representation kan även vara lämpligt för nybörjare eller oinvigda personer som ännu inte lärt sig en notation. För experter kan dock vissa texter bli redundanta om notationen i övrigt uppfyller Moodys principer om intuitiv semantik och framträdande särskiljning.

För att denna princip ska kunna tillämpas så bra som möjligt finns ett behov av att ytterligare tydliggöra hur och när text är lämpligt att använda som komplement till bilder.

Graphic Economy

Moody anger att en notation inte bör innehålla allt för många symboler, då symbolernas betydelse inte går att minnas. Detta gäller speciellt för noviser. Användningen av för många symboler leder till behov av teckenförklaringar eller i värsta fall att läsaren inte förstår vad modellen representerar.

Moody ger fyra olika förslag på lösningar för att hantera Graphic economy, men inget resonemang kring när de olika lösningarna är lämpliga samt hur de relaterar till varandra. Mer förklaring behövs här för att få tillräckligt med stöd vid tillämpning av principen.

Cognitive Fit

I principen Cognitive fit, vilken handlar om att anpassa modellen till målgrupp och uppgift, anger Moody att en notation bör innehålla olika visuella dialekter. Visuella dialekter möjliggör användning för olika målgrupper och uppgifter. Han tydliggör inte hur dessa dialekter bör se ut och byggas upp och relatera till varandra, något som behövs för att kunna tillämpa principen.

Moody ser visuella dialekter som en lösning för förbättring av en notation. Men vid tillämpning av principen på en modell är det så mycket i kontexten som påverkar behoven av anpassningar av representationen av en modell. Ett alternativ till Moodys lösning vid tillämpning på en modell är då att istället för olika dialekter ha *en* uppsättning symboler i notationen, men med mer detaljerade subtyper till de olika symbolerna (exempelvis en symbol för ”roll” och ett antal subtyper för olika typer av roller, så som ”ansvarig roll”, ”utförande roll”). Genom det kan de mer generella, enklare symbolerna användas för modeller som noviser ska förstå och de mer detaljerade, specificerade symbolerna användas för experter.

Moody diskuterar vikten av att förstå de olika behoven och förutsättningarna hos noviser och experter, men en aspekt som ofta kan påverka behoven lika mycket är kontexten. Även om målgruppen är noviser så kanske mer ”komplexa” symboler kan tillämpas om modellen visar en välkänd domän för noviserna, och det är ett fåtal diagram att ta del av.

Moody nämner inte möjligheten att representera en och samma modell på olika sätt gentemot noviser och experter, vilket det kan finnas behov av. Problemet som då uppkommer är dels att behöva underhålla flera diagram som ska representera samma innehåll, samt att se till att de olika abstraktionsnivåerna inte kan tolkas allt för olika.

Problemet är sannolikt inte så stort vid handritning, till exempel vid workshops framför en whiteboard eller post it-lappar på byggplast. I workshopen är alla delaktiga i diskussionen och har kunskap om kontexten, och diskuterar fram en lösning som workshopledaren sedan ritar på tavlan, så när det ritas vet deltagarna vad det är som ska fångas. Då gör det inte så mycket om symbolen blir lite luddig, för det kommer efter workshopen ändå att renritas i ett verktyg. Man ska inte behöva ta hänsyn till att det ska finnas en målgrupp som utan delaktighet i diskussion ska kunna komma in i rummet och förstå den handritade modellen på tavlan

3.7.2.2 Reflektioner kring tillämpning av principerna

Att skapa modeller och diagram som är kognitivt effektiva, och därmed kan nå sitt syfte genom att enkelt kunna förstås av sin målgrupp är mycket viktigt men samtidigt mycket svårt. De principer som Moody har tagit fram utgör kärnan i framställandet av kognitivt effektiva modeller och han menar att principerna kan användas för att utvärdera, jämföra och förbättra existerande visuella notationer, likaväl som att skapa nya (Moody, 2009). Dock avgränsar Moody sitt arbete till notationsnivån och undviker diagramnivå och semantiska frågor (metamodellen). Att använda Moodys principer för att försöka påverka en notation, exempelvis UML, till det bättre är ett svårt arbete och det skulle dröja innan de positiva effekterna kan nå ut i verksamheter. Genom att tillämpa Moodys principer på diagramnivå kan dock positiva effekter genom förbättrade modeller nås snabbare.

Eftersom Moodys principer inte är konstruerade för diagramnivå, krävs det arbete för att ta fram en tillämpning av principerna. En sådan tillämpning kan ge en tydlig vinst och nytta för Försvarsmakten vid både skapande och granskning av diagram och modeller. Förutom vidare arbete med principerna tillkommer det ett ytterligare antal aspekter att ta hänsyn till när de ska tillämpas på modeller, vilket Moody inte berör. Nedan beskrivs några sådana aspekter.

Vid en tillämpning av Moodys principer på diagramnivå bör det ske på hela modellen (som kan bestå av flera diagram), för att kunna identifiera inkonsekvenser som bara blir tydliga mellan diagram. För att underlätta för läsaren bör diagram i en modell vara konstruerade på liknande sätt. Dessa aspekter kan inte fångas genom att enbart läsa ett diagram ur en modell, utan hänsyn måste tas till hela modellen.

En effekt av att Moody inte tar med diagramnivån i sin teori är att den därmed inte innehåller riktlinjer kring estetiska aspekter av ett diagram. Med detta menas bland annat hur linjer bör konstrueras, användning av korsande linjer eller långa kringliggande, eller hur diagrammets hela yta utnyttjas och hur symbolerna distribueras på ritytan. Riktlinjer för detta bör här hämtas från designteorier.

Moodys teori omfattar den visuella notationen, det vill säga symboler och linjer, men en modell kan, förutom själva diagrammen, även kan innehålla matriser, grafer och textuella beskrivningar. Det är därför viktigt att ta hänsyn till även dessa när det gäller att avgöra hur information lämpligast ska representeras i modellen för att nå mesta möjliga effektivt för läsaren.

För att lyckas konstruera kognitivt effektiva modeller behöver modellens mål, syfte och målgrupp vara identifierade och kända för den som skapar eller granskar modellen. Vilka lösningar som är bäst lämpade inom de olika principerna beror nästan alltid på modellens kontext, där även gruppens kunskapsnivå, både avseende notationen och den verklighet modellen representerar, är av vikt. Det är inget självändamål att skapa modeller som är

förståeliga för en slumpmässigt utvald person på gatan - en viss kunskapsnivå kan förutsättas när det gäller läsarnas förståelse för modellens ämnesområde, syfte och mål.

Förutom att ta hänsyn till målgruppen måste även hänsyn tas till hur modellen ska förmedlas. Ska den tas fram för en presentation där åhörarna får 10 sekunder på sig att ta till sig en bild, eller tas den fram för ett långsiktigt arbete där läsarna kontinuerligt får titta på modellen och tänka till. Notera att det även kan vara så att en modell eller ett diagram som ska presenteras kort skulle kunna göras om till en ”presentationsvy”. Exempelvis kan en modell innehållande tre resurselement, en brandbil, en polisbil och en ambulans som har samma symboler, ritas om till en presentation. Vid presentationen kan faktiska bilder på en brandbil, en polisbil och en ambulans användas för att publiken snabbare ska uppfatta bilden. Det görs därmed skillnad på presentationsmaterial och modellen.

Även om notationen ger ett bra stöd för att skapa kognitivt effektiva modeller kan aldrig modellerarens kompetens nonchaleras. Mycket av arbetet på diagramnivå måste hanteras av modelleraren och dennes förmåga att göra rätt bedömningar och avvägningar är av stor vikt. Ett exempel på bedömning är exakt var gränsen går för vilken detaljnivå modellen behöver vara på i de olika diagrammen – räcker det att visa bokningsförfarandet, eller bör även aktiviteter vid avbokning visas? Vad modelleraren lyckas åstadkomma beror även på vilket stöd han/hon får av modelleringsverktyget. Verktyg är ofta konstruerade för att hantera flera olika notationer och blir i viss mån generella. Olika verktygsimplementatörer har även olika lösningar vilket gör att alla verktyg inte ger stöd för att till exempel skapa nya symboler.

Moody har avgränsat sina principer ifrån den semantiska delen av en notation. Dock anger han vid några tillfället att lösningen på vissa problem ligger i de semantiska frågorna. Arbetet med att skapa kognitivt effektiva modeller kan således inte endast lösas genom att se till en notations syntax utan i vissa fall måste de semantiska frågorna omhändertas. Exempel på detta är lösningar i principen Semiotic clarity, där det handlar om att ta beslut om vilka begrepp som ska kunna visas med symboler i notationen.

Inom Försvarsmakten är modeller ett vanligt beskrivningssätt och används bland annat för att beskriva processer, organisationer och verksamheter. Modeller blir dessutom allt vanligare till följd av att Försvarsmakten sedan en tid tillbaka antagit ett modellbaserat angreppssätt för sin förmågeutveckling . För att få ut mer effekt av de modeller som skapas är det därför av vikt för Försvarsmakten att kunna producera och granska modeller för att säkerställa att de uppnår sitt syfte.

4 Diskussion

Att utveckla ledningssystem av tillräckligt hög kvalité är ingen trivial aktivitet, utan ställer stora krav på dem som ska genomföra detta (Kasser, 2007). Dels handlar det om att bygga *rätt system*, dels om att *bygga det på rätt* sätt (Arthur, 1992). Att bygga rätt system är starkt förknippat med att genomföra en adekvat kravhantering (Young, 2001). Viktigt är även att projekt lyckas skapa spårbarhet mellan krav och de funktioner som finns i systemet (Gotel & Finkelstein, 1994). Validering och verifiering är också viktiga delar i systemutveckling, och helt nödvändiga då kravhanteringen och designen av system sällan fungerar optimalt (Hallberg, Pilemalm, Westerdahl, et al, 2008). Utfördes kravhanteringen och designen optimalt skulle både validering och verifiering vara onödiga aktiviteter. Ju mer bristfälligt kravhanteringen och designen genomförs ju mer resurser måste läggas på validering och verifiering samt på omarbetningar. Att bygga system på rätt sätt innebär att vara effektiv när det gäller resursanvändning och att bara genomföra sådant som bidrar till att systemet blir rätt. Möjligheten att återanvända redan kvalitetssäkrade komponenter är möjlighet att erhålla resurseffektivare utveckling.

I denna rapport har sju ansatser för att öka kvalitén i utvecklingen av system beskrivits. Dessa är (1) Lean, (2), Software Product Line Engineering (SPLE) (3) Six Sigma, (4) Capability Maturity Model Integration (CMMI), (5) Goal Oriented Requirements Engineering (GORE), (6) Kvalitetssäkring av krav och (7) The physics of notations. Samtliga dessa ansatser kan vara till nytta för Försvarsmakten, genom att anammas som helhet eller i vissa fall endast tillämpa delar av ansatsen.

Lean syftar till att identifiera och åtgärda aktiviteter och väntetider i utvecklingsprocessen som inte bidrar till utvecklingen av systemet och inte bidrar till systemets kvalité (Oppenheim, 2011). Lean är en väl genomarbetad ansats för att åstadkomma ständiga förbättringar av en verksamhet, men ställer också krav på att verksamheten har förmåga att anamma ett felsökande utan att peka ut syndabockar. Organisationer som lyckas med införandet har dock mycket att vinna i kortare ledtider och effektivare utnyttjande av resurser.

Software Product Line Engineering (SPLE) syftar främst till att stödja återanvändning i bred bemärkelse inom mjukvaruutveckling (Northrop, Clements et al. 2007). Även om SPLE är framtaget för mjukvaruutveckling finns det många delar som ligger i linje med Försvarsmaktens intresse avseende förmågeutveckling, med möjligheter att utnyttja befintliga komponenter för att skapa anpassade förmågor och ledningssystem.

Six Sigma bygger på många års forskning och utveckling kring att mäta och följa upp kvalité (Ficalora & Cohen, 2010). Erfarenheter har dock visat att det inte är lätt att införa Six Sigma, utan det ställer stora krav på organisationen. För

Försvarsmakten är det snarare bättre att vid behov anamma delar av Six Sigma än att ge sig på något storskaligt införande.

Capability Maturity Model Integration (CMMI) erbjuder ett standardiserat sätt att bedöma hur väl organisationer lyckas med utveckling. Att införa metriker så som CMMI erbjuder är resurskrävande och innan detta görs måste det fastställas vad som önskas uppnås. CMMI skulle dock kunna ligga till grund för ett fortsatt arbete om behovet föreligger hos Försvarsmakten.

Goal Oriented Requirements Engineering (GORE) är den ansats inom kravhantering för mjukvaruområdet som är den överlägset mest omskriva de senaste åren (Van Lamsweerde, 2009). Mycket utveckling av metoder, notationer och ramverk inom kravhantering tar idag sin utgångspunkt från GORE. Dock är tillämpningarna av GORE inte alls lika vanliga utanför mjukvaruområdet. GORE kan upplevas som relativt komplext men samtidigt kan erfarenheter av GORE-baserade metoder och notationer vara av värde för Försvarsmakten.

Arbetet med metoden för kvalitetssäkring av krav har kommit längre än arbetet kring övriga ansatser som också beskrivs i denna rapport, då denna genomgått en första utvärdering och revision (Hansson, Granlund, Hallberg, et al., 2010). Utvärderingen som beskrivs i rapporten visar att metoden fungerar som ett stöd för att hitta och delvis korrigera formen som krav är skrivna på. Det är samtidigt slöseri av resurser att felaktigheter gällande kravformulering introduceras i kravspecifikationer överhuvudtaget. Detta medför att vidare arbete bör beakta möjligheten att uppföra stöd för att skriva krav uttryckta i en korrekt form från början. Mer tid kan då läggas på att granska om kraven är innehållsmässigt korrekta, istället för att granska dess form. Inom en överskådlig framtid kommer det dock att finns behov av metoder för att validera formen krav är skrivna på. Ett nästa steg i utvecklingen av metoden är att studera hur denna skulle kunna integreras i Försvarsmaktens utvecklingsverksamhet (Hansson, Granlund & Hallberg, 2011).

The physics of notations beskriver principer för hur notationer ska anpassas för att skapa kognitivt effektiva modeller (Moody, 2011a). Det är viktigt att de modeller som skapas utgör ett effektivt stöd i utvecklingen. Brister i kognitiv effektivitet bidrar till att modellernas informationsinnehåll missförstås vilket kan leda till felaktigheter i slutliga system. Brister i kognitiv effektivitet medför även att förmedla modellerna blir ineffektivt då det tar längre tid att sätta sig in i det modellerna avser att representera. Det fortsatta arbetet kring ”the physics of notations” innefattar att omsätta detta till en metodik för att skapa och granska visuella notationer och modeller ur ett kognitivt effektivitetsperspektiv.

Arbetet som presenteras i denna rapport syftar till att beskriva ett antal ansatser som stödjer ambitionen att utveckla system, vilka skulle kunna vara av intresse för Försvarsmakten att anamma. Det vidare arbetet i projektet kommer att innefatta att studera hur dessa och andra kvalitetsansatser kan nyttjas av

Försvarsmakten och att identifiera och föreslå nödvändiga anpassningar och tillämpningar av valda ansatser.

5 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*, *1,* 19-29. Linköping, Sverige.

Alturki, F. & Khedri, R. (2010). A Tool for Formal Feature Modeling Based on BDDs and Product Families Algebra. *13th Workshop on Requirements Engineerng (WER2010)*. Cuenca, Ecuador.

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

Bashar, N. & Easterbrook, S., (2000). Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering.* 35-46. New York, USA: ACM.

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

Bergström, B. (2004). *Effektiv visuell kommunikation: hur man får ett budskap i text, bild, film, form och färg att nå fram.* Värnamo, Sverige: Carlsson Bokförlag

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

Boyd, J. (1987). *A discourse on winning and losing*. Maxwell Air Force Base, AL: Air University. Library Document No. M-U 43947 (Briefing slides)

Braithwaite, T. (1993). *Information Service Excellence Through TQM: Building Partnerships for Business Process Reengineering and Continuous Improvement.* Milwaukee, USA: ASQC Quality Press.

CMMI Appraisal Program. (2010). P*rocess Maturity Profile - CMMI® For Development SCAMPI℠ Class A Appraisal Results 2010 Mid-Year Update* [Presentation slides]. Pittsburgh, PA, USA: SEI, Carnegie Mellon University.

CMMI Product Team. (2010). *CMMI for Development, Version 1.3*. Technical Report CMU/SEI-2010-TR-033. Pittsburgh, PA, USA: SEI, Carnegie Mellon University. Hämtat från http://www.sei.cmu.edu/reports/10tr033.pdf

Deming, W.E. (1986). *Out of the Crisis*. Cambridge, USA: University Press.

Deming, W.E. (1994). *The New Economics: For Industry, Government, Education*. Cambridge, USA: MIT CAES.

Fagan, M.E. (1986). Advances in Software Inspections. *IEEE Transactions on Software Engineering*, *12*(7), 744–751. Piscataway, USA: IEEE Press.

Ficalora, J. & Cohen, L. (2010). *Quality Function Deployment and Six Sigma a QFD Handbook,* Boston, USA: Pearson education

Firesmith, D. (2003). Specifying Good Requirements. *Journal of Object Technology*. *2*(4), 177-188. Hämtad från http://www.jot.fm/issues/issue_2003_07/column7/

Fricker, S. & Glinz, M. (2010). Comparison of Requirements Hand-off, Analysis, and Negotiation: Case Study. 18th IEEE International Requirements Engineering Conference (pp. 167-176). Ieee. doi:10.1109/RE.2010.29

Gibson, D.L., Goldenson, D.R. & Kost, K. (2006). Performance Results of CMMI®-Based Process Improvement. Technical Report CMU/SEI-2006-TR-004. Pittsburgh, PA, USA: SEI, Carnegie Mellon University

Glazer, H., Dalton, J., Anderson, D., Konrad, M. & Shrum, S. (2008). *CMMI or Agile: Why Not Embrace Both!* Technical Note CMU/SEI-2008-TN-003. Pittsburgh, PA, USA: SEI, Carnegie Mellon University. Hämtat från http://www.sei.cmu.edu/reports/08tn003.pdf

Goldstine, H.H. & von Neumann, J. (1963). *Planning and coding of problems for an electronic computing instrument.* In J. von Neumann, A.H., Taub (Ed.), Collected Works Volume V. (pp. 80-151). UK: Pergamon Press. (Original work published 1947)

Gotel, O.C.Z. & Finkelstein, A.C.W. (1994). An Analysis of the Requirements Traceability Problem. In *Proceedings of the First International Conference on Requirements Engineering* (pp. 94-101). Los Alamitos, CA, USA: IEEE Computer Society Press.

Hallberg, N., Haraldsson, J., Lewau, N., Hansson, J., Granlund, H., Sundmark, T. & Nilsson, S. (2011). *Kravhantering: Best practice*. (FOI-R—3264—SE). Linköping: Totalförsvarets forskningsinstitut (FOI).

Hallberg, N., Pilemalm, S., Sparf, M. & Sjödin, L. (2009) *Modellbaserad utveckling: Omvärldsanalys*, FOI Memo 2842.

Hallberg, N., Pilemalm, S., Westerdahl, L., Jungert, E., Eriksson, H., Andersson, L. & Lindmark, F. (2008) *Principer och metoder för systemutveckling*. FOI 2008, FOI-R--2628--SE.

Hansson, J., Granlund, H. & Hallberg, N. (2011). *Att uttrycka krav i materielmålsättningar: Formulera och granska.* (FOI-R--3250--SE). Linköping: Totalförsvarets forskningsinstitut (FOI).

Hansson, J., Granlund, H., Hallberg, N., Pilemalm, S. & Pilemalm, A. (2010). *Kvalitetssäkring vid kravhantering: Granskning av formuleringar* (FOI-R--3070--SE). Linköping,: Totalförsvarets forskningsinstitut (FOI).

INCOSE. (2009). *Definitions*. Hämtat från http://cse.lmu.edu/about/graduateeducation/systemsengineering/incose/definitions.htm

Jakobsen, C.R. & Sutherland, J. (2009). Scrum and CMMI – Going from Good to Great. *Agile Conference*. 24-28 Aug, 2009. Chicago, USA.

Juran, J.M. & Gryna, F.M.J. (1988). *Quality Control Handbook*. New York, USA: McGraw-Hill

Kasser, J.E. (2007). *A Framework for Understanding of Systems Engineering.* Cranfield, UK: BookSurge Publishing.

Kim, C.S., Spahlinger, D.A., Kin, J.M. & Billi, J.E. (2006). Lean health care: What can hospitals learn from a world-class automaker? *Journal of Hospital Medicine*, *1,* 191–199. doi: 10.1002/jhm.68

Kim, J., Park, S. & Sugumaran, V. (2008). DRAMA: A framework for domain requirements analysis and modeling architectures in software product lines. *Journal of Systems and Software*, *81*(1), 37-55. doi:10.1016/j.jss.2007.04.011

Konrad, S. & Gall, M. (2008). Requirements Engineering in the Development of Large-Scale Systems. *16th IEEE International Requirements Engineering Conference*. (pp. 217-222).

Lapouchnian, A. (2005). Goal-Oriented Requirements Engineering: An Overview of the Current Research. Depth Report, Toronto, Canada: University of Toronto.

Liker, J.K. (2004). *The Toyota way: 14 management principles from the world's greatest manufacturer*. New York, USA: McGraw-Hill Professional.

Locher, D. (2011). *Lean Office and Service Simplified: The Definitive How-to Guide.* New York, USA: Productivity Press Taylor and Francis Group.

Mader, D.P. (2002). Frontiers of Quality: Design for six sigma. *Quality Progress*. July, 2002. (pp. 82–86). Hämtad från http://www.sigmapro.de/de/files/200207%20Quality%20Progress%20--%20DFSS.pdf

LAI. (2011). *Overview Lean Advancement Initiative.*MIT, USA. Hämtat från http://lean.mit.edu/downloads/download-document/2468-lai-overview-november-2011-pdf.html

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

Moody, D.L. (2009). The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering. 35*(6), 756-763.

Moody, D.L. (2011a). The Physics of Notations: A Scientific Approach to Designing Visual Notations in Requirements Engineering. *19th IEEE International Requirements Engineering Conference*. Trento, Italy.

Moody, D.L. (2011b). *Why a Diagram is Only Sometimes Worth a Thousand Words: An Analysis of the BPMN 2.0 Visual Notation.* Hämtat från http://www.business.uq.edu.au/sites/default/files/event/supportingDocs/Analysis%20of%20BPMN%202.0%20Visual%20Syntax.pdf

Moody, D.L. & Hillegersberg, J. (2009). Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams. In D. Gasevic, R. Lämmel and E. van Wyk (Eds.). *Software Language Engineering. Lecture Notes In Computer Science*, Vol. 5452., (pp. 16-34). Berlin, Germany:Springer-Verlag.

Nielsen, J. (1993). *Usability Engineering*. San Diego, California, USA: Academic Press.

Norman, D.A. (1998). *The Design of Everyday Things*. London, England: MIT.

Northrop, L.M. & Clements, P.C. (2007). A *Framework for Software Product Line Practice (Version 5.0).* July, 2007. Hämtad från http://www.sei.cmu.edu/productlines/frame_report/index.html

Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering.* (pp. 35-46). New York, USA: ACM.

Oppenheim, B.W. (2011). *Lean for Systems Engineering, with Lean enablers for systems engineering*, Hoboken, New Jersey, USA: John Wiley & Sons.

Oppenheim, B.W., Murman, E.M. & Secor, D.A. (2009). Lean Enablers for Systems Engineering. *Systems Engineering. 14*(1), 29-55. Wiley Periodicals, Inc.

PARS (2011). *Published Appraisal Results (PARS).* SEI, Carnegie Mellon University. Hämtat från http://sas.sei.cmu.edu/pars/pars.aspx

Penn, M.L. & Siviy, J.M. (2003). *Integrating CMMI and Six Sigma in Software and Systems Engineering.* Pittsburgh, PA, USA: Carnegie Mellon University. Hämtat från http://www.sei.cmu.edu/library/assets/SixSigma_SiviyPenn_SEPG 2003.pdf

Poppendieck, M. & Poppendieck, T. (2003). *Lean Software Development An Agile Toolkit*. Addison Wesley Professional

Radnor, Z. & Walley, P. (2008). Learning to Walk Before We Try to Run: Adapting Lean for the Public Sector. *Public Money & Management*. *28*(1), 13-20. doi:10.1111/j.1467-9302.2008.00613.x

Rebentisch, E., Rhodes, D.H. & Murman, E. (2004). Lean Systems Engineering: Research Initiatives in Support of a New Paradigm. *2and Annual Conference on Systems Engineering Research*. April 15-16. Los Angeles, USA.

SCAMPI Upgrade Team (2011). *Standard CMMI® Appraisal Method for Process Improvement (SCAMPI℠) A, Version 1.3: Method Definition Document*. Handbook CMU/SEI-2011-HB-001. Pittsburgh, PA, USA: SEI, Carnegie Mellon University.

Sellier, D., Mannion, M. & Mansell, J.X. (2008). Managing requirements inter-dependency for software product line derivation. *Requirements Engineering*, *13*(4), 299-313. doi:10.1007/s00766-008-0066-4

Shneiderman, B. (1998). *Designing the User Interface*: *Strategies for Effective Human-Computer Interaction*. (3rd ed.). Menlo Park, CA, USA: Addison Wesley.

Silva, C., Borba, C. & Castro, J. (2011). A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines. *Proceedings of the 14th Workshop on Requirements Engineerng (WER2011).* (pp. 395-406) Rio de Janeiro, Brasilien.

Siviy, J., Penn, M.L. & Harper, E. (2005). *Relationships Between CMMI and Six Sigma.* Technical Note CMU/SEI-2005-TN-005. Pittsburgh, PA, USA: SEI, Carnegie Mellon University. Hämtat från http://www.sei.cmu.edu/reports/05tn005.pdf

Shewhart, W.A. (1939). *Statistical Method from the Viewpoint of Quality Control*. The Graduate School of the Department of Agriculture: Washington DC.

Stoiber, R., Fricker, S., Jehle, M. & Glinz, M. (2010). Feature Unweaving: Refactoring Software Requirements Specifications into Software Product Lines. *18th IEEE International Requirements Engineering Conference* (pp. 403-404). Sidney, Australien. doi:10.1109/RE.2010.59

Van Lamsweerde, A. (2009). Requirments engineering: From System Goals to UML Models to Software Specifications.

Werneck, V. M. B., Oliveira, A. P. A., & Leite, J. C. P. (2009). Comparing GORE Frameworks: i-star and KAOS. In *Proceedings of 12th Workshop on Requirements Engineerng (WER2009)*. (pp.15-26) Valparaiso, Chile.

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

Womack J.P. &. Jones D.T. (2003). *Lean Thinking Banish Waste and Create Wealth in Your Corporation*. New York, USA: Free Press.

Young, R. (2001). *Effective Requirements Practices*. Boston, USA: Addison-Wesley.