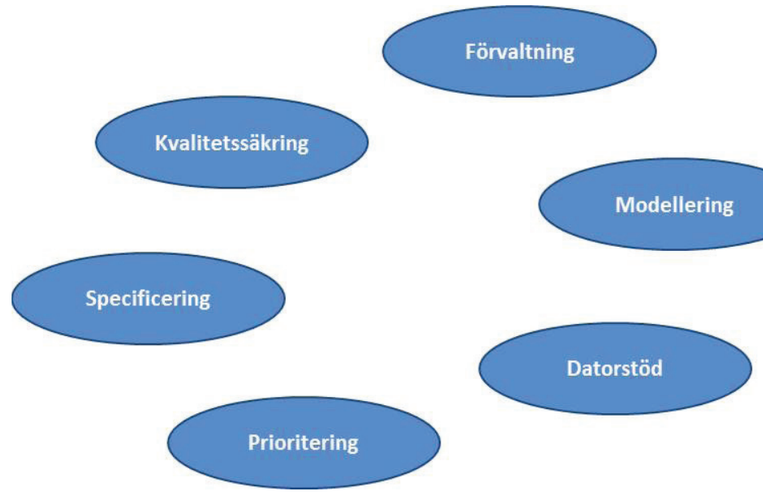
# State-of-the-art: Automatiserad kvalitetssäkring vid kravhantering

HELENA GRANLUND, CHARLOTTE HELLGREN, JONAS HARALDSSON,
THOMAS SUNDMARK, JOACHIM HANSSON, NIKLAS HALLBERG




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
164 90 Stockholm

Tel: 08-55 50 30 00
Fax: 08-55 50 31 00

www.foi.se

FOI-R--3479--SE
ISSN 1650-1942

September 2012

Helena Granlund, Charlotte Hellgren, Jonas Haraldsson, Thomas Sundmark, Joachim Hansson, Niklas Hallberg

# State-of-the-art: Automatiserad kvalitetssäkring vid kravhantering

| | |
|---|---|
| Titel | State-of-the-art: Automatiserad kvalitetssäkring vid kravhantering |
| Title | State-of-the-art: Automated quality assurance within Requirements Engineering |
| Rapportnr/Report no | FOI-R--3479--SE |
| Månad/Month | September |
| Utgivningsår/Year | 2012 |
| Antal sidor/Pages | 36 p |
| ISSN | 1650-1942 |
| Kund/Customer | Försvarsmakten |
| FoT område | Ledning och MSI |
| Projektnr/Project no | E36018 |
| Godkänd av/Approved by | Christian Jönsson |
| Ansvarig avdelning | Informations- och aerosystem |

# Sammanfattning

Väl genomförd kravhantering leder till bättre och mer ändamålsenliga system, men processen att hantera krav är inte trivial och kräver flera olika kompetenser. För att genomföra kravhantering finns många olika ansatser och metoder samt även verktyg för att automatisera delar av processen. Kravhantering kan delas in i ett antal olika aktiviteter, i denna studie ligger fokus på aktiviteterna: *identifiering*, *specificering*, *validering* och *förvaltning* av krav. Rapporten bygger på en litteraturstudie vars mål var undersöka vilka verktyg för automatiserad kvalitetssäkring som är beskrivna i den vetenskapliga litteraturen. Dessa verktyg har även jämförts mot existerade kommersiella verktyg, för att utröna om vetenskapliga verktyg skiljer sig gentemot kommersiella. I studien identifierades 34 verktyg i den vetenskapliga litteraturen och 15 kommersiella verktyg studerades och jämfördes.

Skillnaden mellan de verktyg som beskrivs i litteraturen och de kommersiella studerades beträffande automationsnivå samt vilken aktivitet i kravhanteringen verktygen fokuserar på. Studien visade att de verktyg som beskrivs i litteraturen är mer inriktade på *specificering* av krav och befanns ofta på konceptuell nivå eller prototypstadiet. De kommersiella verktygen hade fler funktioner och fokuserade på aktiviteterna *validering* och *förvaltning*.

Slutsatsen är att de verktyg som beskrivs i litteraturen har ett fokus på stöd för *specificering* och att kommersiella lösningar i hög grad bygger på databaslösningar och främst stödjer projektledning samt *förvaltning* och *spårbarhet*. Verktygstöd för kravhantering syftar i högre grad på att ge stöd till genomförande av aktiviteter inom kravhantering än om kvalitetssäkring av resultatet.

Nyckelord: Kravhantering, litteraturstudie, automatiserade verktyg, kvalitetssäkring

## Summary

A well-implemented requirements engineering process results in better and more efficient systems, but the process of managing requirements is nontrivial and calls for several different competencies/skills. There are several different approaches and methods for requirements engineering, but also tools that automate parts of the process. The process of requirements engineering can be divided into several activities. This study focuses on the activities *elicitation*, *specification*, *validation* and *management* of requirements. This report builds on a literature review of academic articles with the aim of investigating if there are any tools for automated quality assurance. The tools have later on been compared to existing commercial tools, to analyze if they support the same activities in the requirements engineering process. The study identified 34 tools from the academic literature and compared them to 15 commercial tools.

The tools were studied and compared concerning the level of automation and what activities they supported. The study shows that the tools described in the academic literature focused on *specification* of requirements and was at a conceptual level or prototype level, whereas the commercial tools had more functionality and focused on *validation* and *management* of requirements.

The conclusion is that the tools described in the academic literature focus on *specification* and that commercial tools are database solutions that mainly support *requirements management* and *traceability*. Automated tools for requirements engineering is highly focused on implementation rather than on quality assurance.


Keywords: Requirements engineering, literature review, automated tools, quality assurance

# Innehållsförteckning

# 1 Inledning

Att utveckla system är en både dyr och tidskrävande verksamhet. Trots alla ansträngningar som lagts ner, ses många systemutvecklingsprojekt som misslyckade (Söderström 2010; Kasser, 2007). För att nå ett lyckat resultat vid systemutveckling krävs en adekvat kravhantering. Syftet med kravhantering är att avgöra och uttrycka vad system skall klara av samt vilka egenskaper de ska ha. Young (2001) hävdar att upp till 80 % av alla felaktigheter i system kan härledas till en bristfällig kravhantering. Dessa felaktigheter leder till omfattande förseningar i projekt, att projektbudgetar överskrids och till kommersiella konsekvenser såsom förlust av pengar, egendom, personal och i värsta fall i kritiska system leda till förlust av människoliv (Firesmith, 2003). Trots att vikten av god kravhantering sedan länge är känd utförs den till största delen av systemutvecklare med begränsad eller ingen utbildning inom området (Firesmith, 2003). För att stödja kravhanteringen och förebygga brister däri kan verktyg användas för att automatisera delar av arbetet. Dessa verktyg ska bidra till att utföra arbetet både effektivare och med högre kvalitet (Nuseibeh & Easterbrooke, 2000).

Syftet med arbetet som beskrivs i denna rapport är att studera vilka datorstöd som finns för att stödja aktiviteter inom kravhantering. Arbetet är baserat på en litteraturstudie med fokus på automatiserad kvalitetssäkring inom kravhantering samt en översiktsstudie av kommersiella verktyg. Tidigare utvärderingar som gjorts av Försvarsmaktens systemutvecklingsarbeten har visat på problem som relaterar till kravhantering (Hallberg, Pilemalm & Westerdahl, 2008: Eklöf, Hallberg, Hansson, Sjödin & Sparf, 2009). Rekommendationer i dessa utvärderingar uttrycker behov av stöd för kvalitetsarbete vid kravhantering, både i form av metoder och verktyg. Denna studie skall ses som ett bidrag till det fortsatta arbetet med att stödja Försvarsmakten att utveckla sin kravhantering.

Denna rapport baseras på arbete som utförts inom FoT-projektet *Kvalitetsbaserad ledningssystemsutveckling*. Målet med *Kvalitetsbaserad ledningssystemsutveckling* är att utveckla vetenskapligt kvalitetssäkrad och praktiskt förankrad kunskap, samt kompetens avseende kvalitetsbaserad ledningssystemsutveckling som ett stöd för Försvarsmakten.

## 1.1 Frågeställningar

De frågeställningar som ligger till grund för detta arbete är:

- Vad är *State-of-the-art* inom forskningsvärlden när det gäller automatiserade verktyg för kvalitetssäkring vid kravhantering?
- Hur skiljer sig de kommersiella verktygen inom området mot de verktyg som beskrivs i den senaste forskningslitteraturen?

## 1.2 Läsanvisning

Inledningen ger en kort beskrivning av motiv och syfte för studien. Kapitel två, *Bakgrund*, ger en allmän beskrivning av systemutveckling och kravhantering för att ge läsaren förståelse för området litteraturstudien och översiktsstudien omfattar. Kapitel tre, *Genomförande*, beskriver hur litteraturstudien samt översiktsstudien genomförts. Kapitel fyra, *Resultat*, beskriver utfallet av litteraturstudien samt översiktsstudien. Kapitel fem, *Diskussion*, utgör en avslutande diskussion samt slutsatser dragna från resultatet.

# 2 Bakgrund

Detta avsnitt beskriver *systemutveckling* och *kravhantering*.

## 2.1 Systemutveckling

Förutsättningarna för systemutvecklingsprojekt skiljer sig åt avseende exempelvis vilken typ av system som ska utvecklas, tillgång till användarrepresentanter, användarnas tekniska kompetens och tekniska förutsättningar samt hur mycket resurser som avsatts för genomförande. Sedan lång tid har det arbetats med att effektivisera systemutveckling och därtill hitta en universell metod som fungerar för alla typer av utvecklingsprojekt. Att identifiera en universell metod har dock än så länge visat sig inte vara möjligt (Brooks, 1995).

Det finns ett stort antal modeller för att beskriva genomförandet av systemutveckling. Hallberg et al. (2008) menar att systemutvecklingsprocessen generellt kan beskrivas utifrån sju aktiviteter: (1) Kontextanalys, (2) Behovsanalys, (3) Kravhantering, (4) Design, (5) Realisering, (6) Verifiering och (7) Validering.

- Kontextanalys syftar till att analysera den omgivning som det utvecklade systemet ska befinna sig i. Vanliga delar inom kontextanalysen är verksamhetsanalys och intressentanalys, vilka görs för att identifiera och beskriva verksamheter och intressenter.
- Behovsanalys syftar till att identifiera verksamheten och intressenternas olika behov samt dokumentera dessa i en behovsspecifikation. Behov ses som saknad av något som är önskvärt, användbart eller stödjer genomförande av en uppgift alternativt uppnående av ett mål.
- Kravhantering syftar till att specificera vad ett system ska åstadkomma utan att beskriva hur (Siddiqi & Shekaran, 1996). Kravhanteringen syftar till att dokumentera kraven i en kravspecifikation. Kravhanteringen innehåller bland annat aktiviteter som identifiering, analys, specificering och validering av krav (Wiegers, 2003).
- Design syftar till att beskriva hur system ska se ut och fungera och resulterar i en designspecifikation. Designspecifikationer innehåller såväl funktioner som gränssnitt för systemet.
- Realisering syftar till att omsätta den framtagna designspecifikationen till ett system genom utveckling, upphandling av hela system eller inköp och integrering av delsystem.

- Verifiering syftar till att verifiera om designen motsvarar kraven specificerade i kravspecifikationen samt om systemet motsvarar designen.
- Validering syftar till att värdera om systemet ger den önskade effekten för intressenterna, det vill säga om systemet uppfyller intressenternas behov. Ofta innebär valideringen att studera systemet, eller beskrivningar av detta, i den faktiska miljön för systemet för att avgöra hur användarna uppfattar systemet samt hur det fungerar i verksamheten.

Aktiviteterna i ovanstående generella systemutvecklingsprocess kan genomföras på många olika sätt och med olika metoder. Genomförandet av systemutvecklingen kan exempelvis vara iterativ, inkrementell eller evolutionär. Iterativ utveckling innebär att återkoppling sker mellan de ovan nämnda aktiviteterna för att omarbeta eller förfina tidigare arbete. Ett exempel kan vara att kravspecifikationen omarbetas efter att systemet verifierats för att åtgärda de brister som identifierats under verifikationen. En inkrementell utveckling innebär att det utvecklade systemet levereras i etapper där varje leverans ökar funktionaliteten. En evolutionär utveckling är både iterativ och inkrementell.

Inom mjukvaruområdet har agila systemutvecklingsmetoder blivit allt mer populära (Rubin & Rubin, 2011). Enligt Larman (2004) innebär agil utveckling korta iterationer med evolutionär förfining av planer och mål som främjar inkrementella leveranser. Till skillnad mot traditionella systemutvecklingsmetoder försöker agila metoder ha en större flexibilitet och bygger mer på ansikte-mot-ansikte kommunikation och förlitar sig därmed inte på formella dokument i samma utsträckning som traditionella metoder (Cao & Ramesh, 2008). Detta innebär att de ovan beskriva systemutvecklingsaktiviteterna inte är lika tydligt framträdande vid ett agilt arbetssätt och att de dokument som vanligen tas fram i de olika aktiviteterna kan saknas i agila projekt.

Oavsett vilken ansats som används för systemutvecklingen är målet att lyckas med utvecklingen av systemet. Hur lyckad utvecklingen är kan mätas på flera sätt, men Nuseibeh & Easterbrook (2000) hävdar att det främsta måttet på ett systems framgång är till vilken grad det uppfyller sitt faktiska syfte. Emellertid är andelen lyckade systemutvecklingsprojekt låg (Söderström, 2010; Kasser, 2007) och för att övervinna svårigheter vid systemutveckling krävs samverkan mellan olika kompetenser samt en förståelse för användarna och den miljö vari systemet slutligen ska användas.

## 2.2 Kravhantering

Det finns inte någon entydig definition av vad kravhantering innefattar. Enligt Hallberg et al. (2008) är kravhantering ett moment i systemutveckling som

föregåtts av kontextanalys och behovsanalys och som efterföljs av bland annat design och realisering. Detta synsätt delas inte utav alla då både kontextanalys och behovsanalys i vissa sammanhang anses ingå i kravhantering. I ISO/IEC/IEEE 24765 (2010) där kravhantering beskrivs som den vetenskap och disciplin som berör analysering och dokumentering av krav och innefattar de tre aktiviteterna: behovsanalys, kravanalys och kravspecificering. Wiegers (2003) gör en mer detaljerad indelning av kravhanteringen bestående av fyra aktiviter: (1) identifiera krav, (2) analysera krav, (3) specificera krav och (4) validera krav. Därutöver menar Wiegers (2003) att en process för att förvalta krav ingår i kravhanteringen. Nedan ges en kortfattad beskrivning av de aktiviteter som i denna rapport anses ingå i kravhanteringen:

- *Identifiera krav* (eng. Requirement elicitation) syftar till att formulera krav baserat på faktiska behov och innebär ett arbete som utförs i närhet till intressenter, leverantörer och systemets kontext (ISO/IEC/IEEE 24765, 2010).
- *Analysera krav* syftar till att skapa krav med tillräcklig kvalitet. Det innebär att krav förfinas och tydliggörs för alla intressenter, att krav granskas för att identifiera felaktigheter och slutligen att krav prioriteras (Wiegers, 2003).
- *Specificera krav* syftar till att sammanställa och dokumentera krav på ett enhälligt och konsisten sätt.
- *Validera krav* syftar till att validera de specificerade kraven. Kraven valideras för att bedöma om de motsvarar/uppfyller intressenternas faktiska behov och därigenom kan fastställas (Nuseibeh & Easterbrooke, 2000).
- *Förvalta krav* syftar till att underhålla de framtagna kraven under systemets livslängd. Förvaltningen innebär bland annat att versionshantera kraven och omhänderta förändringar i kraven från exempelvis användarna, utvecklarna eller marknaden (Wiegers, 2003).

Förutom att det finns olika synsätt på vad kravhanteringen innefattar finns även ett flertal olika definitioner av vad ett krav är. Siddiqi & Shekaran (1996) definierar ett krav som *vad* ett system ska åstadkomma utan att beskriva *hur*. Sommerville & Sawyer (1997) menar istället att krav kan beskriva problem, såväl som design och begränsningar i lösningar. I standarden ISO/IEC/IEEE (24765:2010, s. 301) definieras krav som ”ett tillstånd eller förmåga som måste uppfyllas eller innehas av ett system, systemkomponent, produkt eller tjänst för att tillfredsställa ett avtal, en standard, en specifikation eller andra formellt tvingande dokument.”

Den senare definitionen är betydligt bredare än de två första och omnämner även kravhanteringens betydelse då kravhanteringens resultat, kravspecifikationen,

ofta fungerar som ett juridiskt dokument som system utvecklas eller upphandlas utefter och verifieras mot för att avgöra om system som levereras uppfyller kraven.

Kravspecifikationer ses ofta som den enda kopplingen mellan kravhantering och övriga aktiviteter inom systemutveckling, eftersom kravhantering ses som en fristående aktivitet (Hallberg et al., 2011). Detta synsätt begränsar möjligheterna för en iterativ utveckling och har ofta lett till dåligt anpassade system. Larman (2004) förespråkar istället att kravhanteringen bedrivs iterativ under hela utvecklingen, vilket innebär att kravhanteringen pågår succesivt och att kraven kontinuerligt kompletteras och modifieras. Ett sådant angreppssätt förebygger fastlåsta kravbilder och dåligt anpassade system. Att ha tillgång till en konkret design eller prototyp kan stödja kravarbetet genom att underlätta förståelsen och kommunikationen kring systemet (Hallberg et al, 2008).

# 3 Genomförande

Projektet genomfördes genom två övergripande aktiviteter; (1) en litteraturstudie och (2) en översiktsstudie av kommersiella verktyg.

Litteraturstudiens och översiktsstudiens syfte var att erhålla en förståelse för state-of-the-art avseende *Automatiserad kvalitetssäkring vid kravhantering* för att i ett ytterligare skede kunna använda denna kunskap för att stödja Försvarsmakten att utveckla deras kravhanteringsprocess.

## 3.1 Litteraturstudie

Litteraturstudien syftar till att undersöka vad som är aktuell forskning rörande verktyg för kravhantering. Studien baserades på publikationer i form av tidsskriftartiklar, konferensbidrag och workshopbidrag.

Utsökningen av litteratur genomfördes genom att (1) identifiera sökord och typ av litteraturkällor, (2) identifiera litteratur baserad på sökord och (3) sortera litteraturen i kategorier för vidare analys.

Utifrån problemområdet och tidigare genomförda studier (Hallberg et al., 2011) identifierades en rad relevanta nyckelord. Dessa användes för att bygga upp följande sökträng:

*("requirements engineering"*
*AND*
*("requirements specification" OR "requirements assessment" OR "requirements review" OR "requirements inspection" OR "requirements quality")*
*AND*
*(tool OR prototype))*
*PUBYEAR > 2004 AND PUBYEAR < 2012)*

Sökningen begränsades till att söka i publikationernas titel, sammanfattning och nyckelord. Sökningen genomfördes under februari 2012 och gjordes i två databaser, *Scopus* och *ScienceDirect.* Båda databaserna är breda indexeringstjänster och inkluderar litteratur från ett flertal utgivare varför de bedömdes täcka ett tillräckligt område för studiens behov.

Sökningarna i *Scopus* och *ScienceDirect* resulterade sammanlagt i 130 publikationer. Den första gallringen av publikationer genomfördes utifrån artiklarnas titel och sammanfattning. Syftet med gallringen var att identifiera artiklar som inte var relevanta för studien och eliminera dessa. Efter gallringen återstod 54 artiklar.

De återstående artiklar studerads i sin helhet och möjliga kategorier och frågor för den fortsatta analysen identifierades. Under denna analys identifierades

artiklar som inte var på engelska, som inte gick att få tillgång till i sin helhet eller som av annan anledning inte bedömdes vara relevanta för studien. Dessa artiklar exkluderades och det slutliga antalet artiklar var 34. De identifierade kategorierna var: *identifiering, specificering, validering* och *förvaltning*. Resultatet av litteraturstudien är beskriven utifrån dessa kategorier och redovisas i resultatkapitlet 4.1, *Litteraturstudie*.

## 3.2 Översiktsstudie av kommersiella verktyg

Syftet med översiktsstudien var att få en övergripande uppfattning om hur väl användningsområdena för kommersiella verktyg överensstämmer med det som beskrivs i den vetenskapliga litteraturen. Översiktsstudien av kommersiella verktyg baserades på artikeln *Requirements Engineering Tools* (Carrillo de Gea et al., 2011) som utvärderar 37 verktyg för kravhantering utifrån tre scenarier. Carrillo de Gea et al. (2011) betygsatte verktygen utifrån deras prestation inom åtta olika kategorier; (1) *Identifiering*, (2) *Analys*, (3) *Specificering*, (4) *Modellering*, (5) *Verifiering och validering*, (6) *Förvaltning*, (7) *Spårbarhet* och (8) *Övriga förmågor*. Prestationen värderades i fem olika nivåer: *very low*, *low*, *medium*, *high* eller *very high (ibid.)*.

I den översiktsstudien som beskrivs i denna rapport studerades de 15 verktyg som fått högst betyg i utvärderingen av Carrillo de Gea et al. (2011). Resultatet av analysen av de kommersiella verktygen beskrivs i resultatkapitlet 4.2, *Översiktsstudie av kommersiella verktyg*.

# 4 Resultat

I detta avsnitt presenteras resultatet av litteraturstudien och översiktsstudien. Vid analysen av de båda studierna kategoriserades verktygen i kategorierna *Identifiering*, *Specificering*, *Validering* och *Förvaltning*. Sammanlagt granskades 34 vetenskapliga artiklar och 15 kommersiella verktyg. Tabell 1 visar antalet verktyg som kategoriserades i respektive kategori. Verktygen är kategoriserade i den kategori som anses vara fokus för verktyget, dock återfinns några verktyg i flera kategorier vilket oftast är fallet för de kommersiella verktygen.

Tabell 1. Fördelning av fokus hos verktyg identifierade i vetenskaplig litteratur och den kommersiella översikten.

| | Identifiering | Specificering | Validering | Förvaltning |
|---|---|---|---|---|
| Litterturstudie | 6 | 16 | 6 | 6 |
| Kommersiella | 9 | 4 | 10 | 13 |

## 4.1 Litteraturstudie

Resultatet av litteraturstudien redovisas utefter kategorierna: *Identifiering*, *Specificering*, *Validering* och *Förvaltning*. Fokus för studien har varit verktyg som kan stödja och kvalitetssäkra kravhanteringen via automatisering. Bland de verktyg som identifierats i litteraturstudien är vissa inte färdigutvecklade. Detta har medfört att visa av dessa verktyg existerar enbart i form av prototyper eller på konceptnivå där automatiseringen är under utveckling.

### 4.1.1 Identifiering

Att identifiera krav inför utveckling av system innebär ett arbete som utförs i närhet till intressenter, leverantörer och systemets kontext. Identifieringen syftar till att formulera krav baserade på de faktiska behoven (ISO/IEC/IEEE 24765, 2010). Emellertid har intressenter svårt för att tydligt uttrycka sina behov. Behoven kommuniceras oftast i form av berättande beskrivningar av önskade och oönskade systemegenskaper (Alrajeh, Ray, Russo & Uchitel, 2007).

Bland litteraturstudiens slutliga 34 publikationer var det sex som beskrev verktygs- eller metodstöd för området att identifiera krav. Av de sex presenterade stöden berörs: (1) återanvändning av krav vid systemutveckling, (2) identifiering av krav i samband med internettillämpningar och (3) identifiering av säkerhetskrav.

Feja, Witt och Speck (2011) presenterar verktyget *BAM* (Business Application Modeler), vilket stödjer manuell identifiering av krav genom att ge tillgång till

tidigare använda processmodeller, formella krav, samt dokument innehållande informella krav. I dessa kravdokument återfinns behov av tidigare system samlade och på så sätt kan erfarenheter återanvändas vid identifieringen av krav för nya system. Detta är dock inte verktygets egentligen fokus utan det är mer inriktat på att specificera formella krav i form av processdiagram.

Agila arbetssätt kan enligt AlAli och Issa (2011) gynna tidiga aktiviteter i utvecklingsarbetet som identifieringen av krav från användare, men en systematisk dokumentering av deras behov missgynnas. AlAli och Issa (2011) föreslår därför ett automatiserat stöd, som är ett subsystem till *StarUML*, för att förbättra dokumentering vid agil systemutveckling. Stödet innebär att användningsfall från redan utvecklade system organiseras i en databas och används som underlag för identifiering av krav för pågående systemutveckling. Författarna har specifikt studerat den agila metoden eXtreme Programning (XP), samt utnyttjandet av användningsfall (eng. use cases) (Beck, 2001; Kulak, & Guiney, 2003). XP bygger på en nära kommunikation mellan utvecklare och användare, samt små och täta releaser nya system versioner. Fördelen med deras verktyg är att en dokumentering sker parallellt med det agila arbetssättet och författarna menar att en 10-30% tidsbesparing i nya projektarbeten och ett större mått av konsistens, kompletthet och struktur i deras dokumentering uppnåtts med deras verktyg.

Internettillämpningar av tjänster får en allt större betydelse för den sociala samhällsstrukturen. Tjänster som tidigare krävde personlig kontakt med myndigheter och andra officiella organisationer kan nu genomföras av den enskilde individen via webben. Zachos, Maiden och Howells-Morris (2008) presenterar ett verktyg som utifrån en påbörjad kravspecifikation för en ny webbtjänst identifierar liknande och redan existerande webtjänster. Information om existerande webbtjänsters funktioner kan sedan användas för att förbättra och komplettera kravspecifikationen för det nya systemet som är under utveckling.

Valderas och Pelechano (2007) berör också internettillämpningar genom ett verktyg baserat runt en kravontologi rörande webbtjänster. En kravontologi beskriver de koncept och konceptrelationer som är relevanta för en specifik typ av system. Genom att ett interaktivt stödverktyg automatiskt ställer frågor baserade på ontologin till systembeställaren kan en systematisk beskrivning av behov uppnås. Beskrivningen som ges transformeras automatiskt till en specifikation som systembeställaren direkt kan validera genom att kontrollera att rätt krav har identifierats från den givna behovsbeskrivningen.

Ett alternativ till att identifiera krav från intressenterna är att återanvända krav för liknade system. Återanvändning av krav ses som ett sätt att reducera kostnader, varpå det finns ett intresse att använda verktyg som kan stödja identifieringen av lämpliga krav från andra domäner och använda dessa för det system som ska utvecklas (Udomchaiporn, Prompoon & Kanongchaiyos, 2006). Udomchaiporn et al. (2006) har utvecklat ett tillvägagångssätt att identifiera relevanta krav från

andra domäner genom att analysera stukturen hos användningsfall. Då relevanta redan färdiga användningsfall identifieras kan de krav relaterade till användningsfallet potentiellt vara återanvändbara. Detta skulle reducera kostnader vid kravspecifikation av mjukvarusystem. Verktyget analyserar inte användningsfallen direkt utan den tillhörande beskrivningen, vilken ofta skrivs i naturligt språk. En beskrivning över ett användningsfall kan enligt Udomchaiporn et al. (2006) delas upp i nio aspekter: namn, objektiv, aktör, relation, förutsättningar, postconditions, normala händelseflöden, subflöden och alternativa/exceptionella händelseflöden. Genom att lagra beskrivningar över användningsfall i en databas kan analytiker söka efter krav enligt dessa nio aspekter. De krav som hittas viktas för att få fram krav som är lämpliga för det nya projektet. Begränsningarna i processen är att verktyget går på syntaktiskt likhet och inte kan uppfatta en semantisk likhet eller skillnad mellan ord.

Ett problem vid systemutveckling är att organisationers säkerhets- och sekretesskrav inte kommer med vid identifiering av krav. En metod för att behjälpa detta är *ReCAPS* (Requirements-based Access Control Analysis and Policy Specification). Metoden består av 32 heuristiker och integrerar säkerhets- och integritetsriktlinjer i kravanalysen och mjukvarudesignen. *ReCAPS* avses resultera i en mer komplett, korrekt och mindre tvetydig dokumentation av säkerhetskrav. Ett försök att automatisera arbete baserat på *ReCAPS* utgörs av verktyget *SPRAT* (Security and Privacy Requirements Analysis Tool) (He & Antón, 2009). Verktyget ger stöd för att analysera och specificera säkerhets- och sekretesskrav, riktlinjer och ACPs (Access Control Policies).

SURE (Secure and Usable Requirments Engineering) är en metod utvecklad av Romero-Mariona, Ziv och Richardson (2010) som ska stödja identifiering av säkerhetskrav. Det är även en process som hjälper till vid kartläggning av säkerhetskrav när mjukvara ska testas. För att automatisera processen utvecklade Romero-Mariona et al. (2010) ett verktyg, *ASSURE* (Automated Support for Secure and Usable Requirments Engineering), som är en webbaserad databas. Med hjälp av SURE får användarna hjälp med att identifiera, analysera, och specificera säkerhetskrav samt *misuse cases* och en dokumentering av hot som skulle påverka projektet. Med hjälp av SURE gör användarna en specificering genom att baserat på säkerhetsutsagor (eng. security statements) identifiera säkerhetsbehov (eng. security needs) som sedan omsätts till säkerhetskrav. Det stöd verktyget *ASSURE* ger är att visualisera skapandeprocessen.

### 4.1.2 Specificering

Specificering av krav innebär att sammanställa och dokumentera identifierade krav. Att skriva en korrekt och konsekvent kravspecifikation är en av de viktigaste uppgifterna inom kravhantering (Garbers & Periyasamy, 2006), men att manuellt omformulera användarnas behov till en kravspecifikation är mödosamt och leder ofta till felaktigheter, motsägelser och extra kostnader

(Dascalu, Fritzinger, Debnath & Akinwale, 2006). En automatisering av processen, kan spara tid och pengar samt att höja kvalitén på resultatet. I litteraturen var det 16 artiklar som beskrev verktygs- eller metodstöd för området att specificera krav. Dessa presenteras under områdena (1) *Naturligt språk*, (2) *Modeller*, (3) *Naturligt språk och modeller* och (4) *Maskininlärning*.

#### 4.1.2.1 Naturligt språk

Kravspecifikationer skrivs vanligtvis i naturligt språk. Naturligt språk är dock inte det mest effektiva när det gäller att specificera krav, eftersom det innehåller tvetydigheter och vagheter som sedan återspeglas i kraven (Mavin, Wilkinson, Harwood, & Novak., 2009). För att komma tillrätta med detta har verktyg som stödjer olika delar av specificeringen av krav utvecklats. Ett sådant verktyg är *QuARS* (Bucchiarone, Gnesi & Pierini, 2005; Lami, Gnesi, Fabbrini, Fusani, & Trentanni, 2005). *QuARS* är ett verktyg som ger kravanalytiker möjligheten att kontrollera krav skrivna i naturligt språk utifrån ett lingvistiskt perspektiv. Verktyget stödjer kravanalytiker i deras arbete genom att identifiera och peka ut svagheter i kravspecifikationer. De defekter som identifieras kan vara av lexikal karaktär och identifierar då subjektivitet, vagheter och svagheter i krav (Bucchiarone et al, 2005). Andra defekter som identifieras är av syntaktiskt karaktär och identifierar då bland annat implicitet eller underspecificerade krav (Bucchiarone et al, 2005). Verktyget stödjer även analytiker genom att tillhandahålla metriker som ger mått på hur välskriven kravspecifikationen är. Dessa mått anges utifrån respektive kontrollpunkt (subjektivitet, vaghet etc.) samt ett sammanhållande värde på hela kravspecifikationen. Att göra en lingvistisk granskning manuellt är en tidskrävande uppgift, vilket kan underlättas med denna typ av verktyg. Verktyget är dock ett hjälpmedel och ska inte ses som en ersättare för en mänsklig granskare (Lami et al, 2005).

Välskrivna kravspecifikationer är viktigt ur flera aspekter, bland annat för att kravspecifikationerna ofta fungerar som ett kontrakt mellan kund och leverantör. De flesta kravspecifikationer skrivs endast med stöd av ordbehandlingsprogram och en stor del av ansvaret på att specifikationen blir korrekt ligger på författarna (Garbers & Periyasamy, 2006). *Napkins* är ett verktyg som ska ge stöd till författaren då kravspecifikation författas och syftar till att ge bättre kravspecifikationer med fokus på innehåll (Garbers & Periyasamy, 2006). *Napkins* stödjer författandet av kravspecifikationer så att dessa konstrueras utifrån standarden IEEE 830-1998. Detta genom att tillhandahålla ett gränssnitt för att mata in krav och klassificera dessa, vilket medför att verktyget automatiskt strukturerar kravspecifikationen. Detta ger en möjlighet att fokusera mer på innehållet i kraven än att behöva fokusera på hur kravspecifikationen ska vara uppbyggd. *Napkins* ger även stöd för att utvärdera kravspecifikationer. För detta används metriker som är framtagna av Software Metrics Program hos NASA. Utvärderingen handlar om att verktyget identifierar olika typer av svagheter i kravspecifikationen som vaga uttryck, till exempel *adekvat* eller

*lämpligt*, eller då det finns valmöjligheter i krav, till exempel då ord som *kanske* eller *valbart* har använts. Verktyget kan även värdera komplexiteten i kravspecifikationen och utifrån det ge en uppskattning på hur lång tid det specificerade systemet skulle ta att implementera (Garbers & Periyasamy, 2006).

Ett verktyg för att motverka att lingvistiskt felaktiga krav inarbetas i kravspecifikationer är *HeRA* (Knauss, Schneider & Stapel, 2009). *HeRA* stödjer kravanalytiker att lära sig att skriva bättre krav genom kontinuerlig feedback, bland annat om kraven är tvetydiga och ofullständiga. Genom en kontinuerlig utvärdering av krav ska analytikern direkt bli uppmärksammad på defekter i de krav de skrivit. Detta är även tänkt att tillföra organisationer en bättre kunskap om hur krav formuleras.

Att skapa kravspecifikationer för mjukvarusystem med både funktionella och icke-funktionella krav utifrån naturligt språk kräver mycket av kravanalytikerna. För att effektivisera processen skapades *NALASS* – Natural Language Syntax and Semantics (Georgiades & Andreou, 2010). *NALASS* är ett verktyg som automatiserar metoden NLSSRE (Natural Language Syntax and Semantics Requirements Engineering). Ett större ansvar läggs på användarna, som själva får svara på fördefinierade frågor som skapas av analytikern med hjälp av en guide som finns i verktyget. Utifrån svaren på frågorna skapar verktyget diagram samt ett kravdokument bestående av formaliserade meningar.

#### 4.1.2.2 Modeller

Ett stöd för att gå från informella användarkrav till formella kravspecifikationer är det grafiska verktyget *W_PSC* (Wizard Property Sequence Charts) (Autili & Pelliccione, 2008). Verktyget utgör en guide som leder fram till beslut som eliminerar oklarheter och dubbeltydigheter när användarkrav görs om till så kallade Property Sequence Charts (PSC) scenarier som i sin tur resulterar i en kravspecifikation.

För att minimera risken för fel vid specificering av funktionella användarkrav har Siqueira & Silva (2011) tagit fram det semi-automatiska verktyget, *EMUCase*, som skapar grafiska Enterprise modeller. Med hjälp av verktyget får användarna själva skapa modeller över hur verksamheten fungerar idag, och hur de vill att det ska fungera i framtiden. Verktyget transformerar sedan en abstrakt syntax till en konkret syntax i en användningsfallsmodell (eng. use case model). En verksamhetsmodell gör det möjligt att representera både krav och domänkunskap och till hjälp för att skapa dessa modeller finns en guide med elva heuristiker, där heuristikerna stödjer transformationen av syntaxen för att eliminera tvetydigheter, men inte med att förfina kraven (Siqueira & Silva, 2011).

#### 4.1.2.3 Naturligt språk och modeller

Ett sätt att reducera antalet felaktigheter i kravspecifikationer är att genom olika typer av granskningar identifiera och åtgärda dessa. En annan ansats är att

förhindra att felaktigheter kommer in i kravspecifikationer från början, exempelvis genom att tillhandahålla mallar för krav. Ghazel (2011) beskriver verktyget *TGM* som bland annat tillhandahåller mallar för att skriva temporala krav, med temporala krav avses krav på händelsers inbördes ordning och tidsapsekter på deras relationer. Mallarna stödjer kravanalytiker att skriva krav på ett enkelt och precist sätt (Ghazel, 2011), och på så sätt undviks vanliga fel som tvetydighet och inkonsekvens. Förutom att tillhandahålla stöd vid formulering av krav stödjer TGM identifiering av inkonsekvens i kravspecifikationen genom ett grafiskt gränssnitt. Genom att visuellt lyfta fram krav tydliggörs motstridigheter hos temporala krav.

Även om en kravspecifikation är skriven i naturligt språk kan en granskning göras baserat på annat format. Verktyget *MaramaAI* (Kamalrudin, Hosking & Grundy, 2011) används för granskning av kravspecifikationer utifrån konsistens, kompletthet och korrekthet genom att först transformera den skrivna specifikationen till användningsfall. Granskningen genomförs således inte på de skrivna kraven utan på användningsfallen. Genom att jämföra tidigare framtagna användningsfall med det specifika projektets kan jämförelser såsom inkonsekvens, ofullständighet och felaktigheter identifieras och visuellt tydliggöras. Verktyget försöker inte att korrigera dessa felaktigheter utan tydliggör endast dessa som problem och sedan är det är upp till kravanalytikern att hantera detta.

Vid utvecklingen av produktfamiljer, applikationer som har vissa gemensamma funktioner, kan en kärnspecifikation användas. Kärnspecifikationen specificerar de funktioner som samtliga produkter baseras på och har gemensamt. Denna kan sedan utvidgas för en specifik del av produktfamiljen, i form av en produktspecifikation. Siy, Aryal, Winter och Zand (2007) har skapat en prototyp för kravställning för produktfamiljer. Prototypen hanterar både funktionella och icke-funktionella krav. Inledningsvis transformeras krav skrivna i naturligt språk till ett DSL (Domain Specific Language). Med hjälp av en speciellt utvecklad semantik och syntax för detta DSL möjliggörs att de icke-funktionella kraven kan vävas ihop med de funktionella vilket sedan utgör produktspecifikation. Sammanvävningen gör att en domänexpert kan se hur de icke-funktionella kraven hänger ihop med funktionella kraven inom dennes expertområde. Detta underlättar kommunikationen kring kraven och domänexpertens granskning.

#### 4.1.2.4 Maskininlärning

Maskininlärning är en gren av artificiell intelligens som handlar om design och utveckling av algoritmer som tillåter datorer att lära från empiriska data. Datorn bygger upp kunskap baserad på goda och dåliga exempel. Algoritmen ger sedan datorn förmåga att bedöma nya egenskaper hos det empiriska materialet. Tekniken har prövats på automatiserade system avseende specificering av krav (Alrajeh, Ray, Russo & Uchitel, 2007; Popescu, Rugaber, Medvidovic & Berry, 2008; Umber, Bajwa & Asif Naeem, 2011; Weston, Chitchyan & Rashid, 2009).

Weston et al. (2009), Popescu et al. (2008) och Umber et al. (2011) föreslår lösningar som baserar sig på Natural Language Processing för problem som att identifiera konflikter och tvetydigheter i kravdokument samt för att transformera naturligt språk till formella språk. Transformeringen från naturligt till formellt språk krävs för att kunna hantera kraven algoritmiskt. Alrajeh et al. (2007) behandlar identifiering av mål från scenarier och initiala kravdokument samt att överföra dem till krav med en lösning som baseras på Inductive Logic Programming.

*SRSQAS* (Software Requirements Specification Quality Analysis System) är ett prototypverktyg för att testa kvalitén på kravspecifikationer. *SRSQAS* bygger till viss del på uppskattning från användaren. Användaren får svara på frågor som värderas 1-5, där 1 är minst signifikant. Frågorna berör olika kvalitetsindikatorer och svaren är kopplade till olika kvalitetsattribut för kravspecifikationen. Kvalitetsattributen är elva attribut som en kravspecifikation skall försöka att uppfylla. Kvalitetsindikatorerna viktas och utifrån de angivna svaren beräknas sedan kvalitén på kravspecifikationen. Verktyget föreslår även förändringar som kan förbättra kravspecifikationen. Förändringar baseras på tidigare genomförda analyser och verktyget lär sig av gamla analyser (Jani & Mostafa, 2011).

### 4.1.3 Validering

Att validera krav innebär att validera att specificerade krav är relevanta, korrekta samt att kravspecifikationen är fullständig. Sex publikationer berörde området validera krav. Dessa berör: (1) validering av generiska kravspecifikationer, (2) prototyper, (3) validering av modeller, (4) internettillämpningar och (5) tekniker runt aktivitetsdiagram (eng. activity diagram).

Generiska kravspecifikationer, som används vid utveckling av produktfamiljer, innebär en ökad risk i och med att fel i kraven påverkar flera produkter. Robinson-Mallett, Grochtmann, Köhnlein, Wegener och Kühn (2010) har utvecklat ett tillägg till verktyget Rational Doors, av IBM, för att validera denna typ av krav och förhindra sådan felspridning. Den generiska kravspecifikationen består av generella krav som genom att justera en enskild variabel kan nyttjas för flera olika produkter. Ett exempel är att olika varianter av e-post-system inom samma produktfamilj ska hantera olika mängder trafik per dygn. Istället för att ha flera produktspecifika krav används ett generellt krav som specificeras genom att trafikmängden är en variabel. Sådan användning av generella krav medför en effektivisering av specifikationshantering eftersom det innebär färre unik krav att underhålla. Vikten av att validera kraven är dock högre i och med den nämnda risken att om fel införs i den generiska kravmängden kommer felen att fortplanta sig i hela produktsortimentet.

Generellt är det svårt för intressenter för system att uttrycka och validera behov. Att använda prototyper kan utgöra ett medel för analytikern att tillsammans med

intressenter utföra valideringen. Matos och Sousa (2010) har utvecklat ett system för att stödja validering vid agil utveckling genom automatiserad generering av både testfall och prototyper av användargränssnitt utifrån användningsfall. Testfallen är konstruerade att användas mot prototypen och därmed kan de ursprungliga användningsfallen valideras. Liknande försök görs även av Ogata och Matsuura (2010) men de utgår från en requirements analysis modeling cycle (RA-modell). Denna modell består av aktivitetsdiagram, klassdiagram och objektdiagram modellerade i UML (Unified Modeling Language) med verktyget *Astah*. Med modellen som indata i Ogata och Matsuuras (2010) metod genereras automatiskt en användargränssnittsprototyp. Denna prototyp kan sedan valideras dels av utvecklarna för att kontrollera konsistenesen hos RA-modellen, men även av kunderna för att validera mot deras affärsflöden och behov.

Vid utveckling av informationssystem används ofta affärsprocessmodeller för att specificera krav på mjukvara. Det innebär att kvaliteten på mjukvaran i hög grad påverkas av kvaliteten hos affärsprocessmodellerna (Feja et al. 2011). För att validera att semi-formella modeller överensstämmer med mot informella krav krävs dock manuellt arbete. Med formellt uttryckta krav kan motsvarande processmodeller istället automatiskt valideras. Feja et al. (2011) vill med modelleringsverktyget *BAM* integrera formella och grafiska krav vid modellering. Med verktyget BAM kan formella krav representeras visuellt i form av affärsprocessmodeller. Detta möjliggörs genom att bland annat formella regler definieras vilka processmodellerna följer varför det går att automatiskt validera överenstämmelsen mellan de formella kraven och motsvarande visuell representation i form av affärsprocessmodellerna (Feja et al., 2011).

Aktivitetsdiagram i UML2 ger nya möjligheter enligt Knieke, Huhn och Lochau (2008) att specificera krav och författarna har utvecklat en variant de kallar Live Activity Diagram (LAD). LAD-diagram specificeras enligt en formell syntax och semantik vilket möjliggör att de går att exekvera. Det innebär att krav specificerade i form av LAD-diagram går att automatiskt validera genom att aktiviteten kan simuleras.

### 4.1.4 Förvaltning

Förvaltning av krav innebär att hantera krav, efter det att de identifierats och specificerat, under återstoden av systems livscykel (Nuseibeh & Easterbrooke, 2000). I litteraturstudien var det sex av de 34 publikationerna som berörde verktyg för att hantera förvaltning. Ett verktyg för att stödja hanteringen av de textuella aspekterna i användningsfallsmodellering är *STORM* (Software Tool for the Organization of Requirements Modeling) (Dascalu et al., 2006). Verktyget ska spara tid och minimera fel genom att automatisera och konkretisera uppgiften. Det hjälper till med förvaltningen av krav genom att underhålla användningsfallen och scenarier samt upprätthålla spårbarhet. Verktyget ska även stödja övergången mellan kravspecifikationen och en design, dock stöder

inte verktyget generering utav kravspecifikationen, utan detta måste göras manuellt.

För spårbarhet i kravhantering har Hong, Kim och Lee (2010) initierat ett arbete med ett verktyg för att hantera förändringar och versioner samt länka krav mellan olika delar av systemet. Att veta hur krav relaterar till varandra, var de ursprungligen kommer från och hur de förändras är viktigt för att upprätthålla en tillfredställande och aktuell kravmängd. Genom att analysera likheten mellan dokument eller meningar för att skapa en spårbarhet från krav, har Koo, Seong, Yoo, Cha och Yoo (2005) tagit fram ett verktyg som leder kravspecifikationen vidare till systemdesign och testfall. Krav specificeras och omformuleras ofta i en iterativ process parallellt med eventuella prototyper. Ett problem som kan uppstå är att den formella specifikationen och prototypen hamnar ur fas och börjar differentiera. För att undvika detta och åstadkomma spårbarhet mellan den uppdaterade specifikationen och prototypen skapade Deshmuk och Wadhwa (2007) en metod där prototyp och specifikation kopplas samman via en metamodell.

Monteiro, Ebert och Recknagel (2009) har utvecklat en plattform som stödjer sammarbetet runt krav, mellan globalt distribuerade enheter, under systemets utveckling. I sitt arbete beskriver de hur de löst informationsutbyte i plattformen med Requirements Interchange Format[1]. Arpinen, Hämäläinen och Hännikäinen (2011) har även de uppmärksammat problemet med sammarbete runt hantering av krav under ett systems utveckling. De vill påvisa vikten av att ha en tydlig metamodell för hanteringen. Med en tydlig metamodell kan utvecklingsprojekt, där flera olika kravhanteringsverktyg används, uppnå ett välfungerande samarbete där spårbarhet och andra svårigheter med krav hanteras trots olika arbetssätt hos olika projektdeltagare. Arpinen et al. (2011) redovisar sin metamodell och beskriver hur de gjort en prototyp för metamodellen baserad på UML.

## 4.2 Översiktsstudie av kommersiella verktyg

Carrillo de Gea et al. (2011) delade in kravhanteringsverktygen i sex kategorier: *Identifiering*, *Analysering*, *Specificering*, *Verifiering och validering*, *Förvaltning* samt *Övriga förmågor*. Dessa kategorier, sånär som på *Analysering* och *Övriga förmågor*, passar in i den kategorisering som gjorts inom litteraturstudien. Resultat är dock inte beskrivet i samma struktur som litteraturstudien då dessa verktyg ofta stödjer flertalet av dessa aktiviteter. Resultatet av översiktsstudien av kommersiella verktyg är beskriven utefter varje verktyg där det sist i

[1] Requirements Interchange Format är en XML-baserad standard för att uttrycka information. Formatet utvecklades initialt för tysk bilindustri.

beskrivningen av varje verktyg refereras till vilka kategorier de härrör. En sammanställning av verktygens viktigaste egenskaper återfinns i Tabell 2.

Tabell 2 Sammanställning av kommersiella verktyg.

| | | | Stödjer | | | |
|---|---|---|---|---|---|---|
| **Verktyg** | **Egenskaper** | **Webbaserat** | **Identifiering** | **Specificering** | **Validering** | **Förvaltning** |
| **Acclaro DFSS** | Använder *Six Sigma*-metodik och verifierar att kundens behov har omhändertagits i slutprodukten. | | X | X | X | X |
| **CASE Spec** | Stödjer samarbete mellan grupperingar genom en central databas. Skapar dokument automatiskt med stöd av funktioner för filtrering, gruppering och sortering | | | | X | X |
| **Cognition Cockpit** | Importera information från Microsoft Office, Matlab och DOORS. Stödjer hela produktutvecklingen. | X | X | X | X | X |
| **Cradle** | Ger möjlighet till distribuerad och webbaserad kravhantering. | X | X | X | X | X |
| **GMARC** | Kontrollerar kravspecifikationen mot aspekterna sammanhang, konsekvens, kompletthet och korrekthet. | | X | X | X | X |
| **inteGREAT** | Automatisk generering av kravdokument och ett antal olika diagramtyper. | | X | X | X | X |
| **IRqA** | Ger spårbarhet genom hela projektets livscykel. Stödjer återanvändning från tidigare projekt. | | | | X | X |
| **MKS integrity** | Verktyg för mjukvarusystems hela utveckling och underhåll. | | X | X | X | X |
| **PACE** | Stödjer samarbete och kommunikation genom bättre delning och tillgång till krav och förändringar. | X | X | X | X | X |
| **Polarion Requirements Management** | Interagerar med Microsoft Word för skapa och redigera krav i textdokument. | X | X | X | X | X |
| **Psoda** | Ger översikt av de krav som ännu inte validerats. | | X | X | X | X |
| **QPack** | Ger projektledaren överblick av upparbetade krav. | | X | X | X | X |
| **ReqMan** | Krav lagras i en databas och redigeras i textdokument. | | X | X | X | X |
| **Reqtify™** | Stödjer projektledningen med konsekvensanalyser och spårbarhet. | | X | X | X | X |
| **TraceCloud** | Integrerat med Microsoft Office. Projektledaren kan kontrollera vilka personer som får utföra vad. | X | X | X | X | X |

### *4.2.1* Verktygsbeskrivningar

*Cognition Cockpit* är ett web-baserat verktyg som kan importera information från Microsoft Office, Matlab samt andra kravverktyg så som exempel Doors (Cognition Corporation, 2012). Verktyget är tänkt för hela produktutvecklingen och kan användas av flera olika användargrupper så som projektansvariga (managers), olika utvecklingsteam och PR-team (Cognition Corporation, 2012). Verktyget hanterar även spårbarhet mellan kraven och parametrarna som styr kraven (Alexander, 2012). Det är ett av de mest kapabla verktygen när det gäller både kravidentifiering samt verifiering och validering (Carrillo de Gea et al., 2011). Kraven ses inte bara som text utan behandlas som multidimensionella dynamiska objekt som exempelvis kan associeras till risker och kostnader (Cognition Corporation, 2012).

*GMARC* (Generic Modeling Approach to Requirements Capture) är främst en metod för kravhantering vilken understöds av ett verktyg med samma namn. Metoden grundar sig på att maximera kraven utifrån aspekterna sammanhängande (eng. coherent), konsekvent (eng. consistent), komplett (eng. complete) och korrekt (eng. correct) och verktyget har stöd för att kontrollera kravspecifikationer mot dessa mål. Identifiering av krav stödjs av verktyget genom att användarna får besvara frågor. Verktyget lär sig även av detta för varje projekt och har dessutom stöd för bearbetning av naturligt språk. Dessutom stödjer verktyget delvis automatisk transformation av kraven till diagram, som sedan kan simuleras och därmed valideras (Computer System Architects Ltd., 2002).

*Acclaro DFSS* är ett kravhanteringsverktyg utvecklat av Axiomatic Design Solutions, Inc. DFSS-delen av namnet står för ”Design for Six Sigma”, vilket är en metodik för att optimera verksamhetsprocesser. Fokus hos verktyget ligger på identifiering av kundernas behov via så kallade kundrösttabeller (eng. Voice-of-customer table), samt verifiering av att dessa verkligen har omhändertagits i slutprodukten. Verktyget stödjer huvudsakligen identifierings- samt analysfaserna av kravhantering (Axiomatic Design Solutions, Inc., 2006).

Verktyget *CASE Spec* är fokuserat på att skapa och validera systemspecifikationer. Det har även funktioner för att skapa diagram enligt en rad olika notationer. Verktyget stödjer spårbarhet till exempel genom att länka samman delar av användningsfall med testfall, felrapporter och uppgifter. Samarbete stöds genom att ge tillgång till en central databas, där olika former av dokument automatiskt kan skapas med hjälp av funktioner för filtrering, gruppering och sortering. Verktyget stödjer även framtagandet och användandet av användningsfall samt att det går att skapa relationer mellan textuella specifikationer och diagram (Goda Software, 2012).

*QPack* är ett kravhanteringsverktyg vars syfte är att ge projektledare bättre översikt när det gäller uppfyllnadsstatus för de krav som ingår i projektet.

Verktyget ger dessutom stöd för att planera, godkänna, spåra samt validera krav. Det finns även en speciell utgåva av verktyget specifikt designad för hantering av krav inom läkemedelsindustrin. Verktyget stödjer validering- och förvaltningsfaserna inom kravhantering (Orcanos, 2009).

Verktyget *Psoda* syftar till att ge bättre översikt över vilka krav som ännu inte har validerats. Verktyget stödjer därmed både valideringss- och förvaltningsfasen inom kravhantering (Psoda, 2011).

*Reqtify*™ är ett kravverktyg för projektledningen. Det hämtar redan upparbetad information från företagets servrar och tillhandahåller information om kraven för att kunna utföra konsekvensanalyser och hantera spårbarhet. Verktyget går att använda tillsammans med andra kravhanteringsverktyg, ordbehandlare, projektledningsverktyg och ”product life cycle management tools” men verktyget stödjer i sig självt utvecklings- och verifierings och valideringsprocessen (Greensoft, 2010).

*MKS integrity* är ett kravhanteringsverktyg för mjukvaruutveckling (SSLM-Software System Lifecycle Management). *MKS integrity* behandlar fyra delar av SSLM: kravhantering, testhantering, konfiguration- och ändringshantering samt förvaltning av systemmodeller (The Product Development Company, 2012). Verktyget är byggt för att vara en del av ett större projektstödsystem (Alexander, 2012). Verktyget organiserar kraven hierarkiskt, har spårbarhet till källkoden och går att integreras med Microsoft Word (Alexander, 2012).

*Polarion Requirements Management* (Polarion) är ett webbaserat stödverktyg för att stödja hanteringen av krav. Systemet stödjer import av Word-dokument där enskilda krav identifieras enligt regler baserade på syntax eller formatering. Verktyget fortsätter dock presentera kraven i ett läsbart dokumentformat. Det möjliggör att kraven kan skrivas och editeras som ett sammanhängande dokument, medan de samtidigt är kopplade till en databas. Spårbarhet stöds genom att det är möjligt att i verktyget se vem som ändrat ett krav, på vilket sätt och varför. Verktyget stödjer även samarbete via till exempel diskussionsforum, vilket kan användas för att dokumentera hur beslut tagits. Krav kan kopplas till testfall och verktyget bibehåller spårbarheten däremellan (Polarion Software, 2012).

*Cradle* är ett kravhanteringsverktyg som utvecklats av företaget 3SL. Verktyget fokuserar på spårbarhet och versionshantering av krav samt möjligheterna till distribuerad och webbaserad kravhantering. *Cradle* kan användas för allt ifrån enstaka små projekt till en hel flora av större utvecklingsprojekt. Verktyget stödjer kravarbetet i identifiering-, analys- och förvaltningsfaserna (3SL, 2012).

*TraceCloud* är ett kravhanteringsverktyg som stödjer hela kravprocessen. Det är ett web-baserat program med Microsoft Office integration där projektledaren kan kontrollera vilka roller som kan och får göra vad med kraven. Verktyget stödjer identifiering, analys, specificering, validering och förvaltning av krav

(TraceCloud, 2010). Men verktyget har visat sig främst bra för validering och verifiering samt kravspecificering (Carrillo de Gea et al., 2011).

*IRqA* är ett verktyg för specificering och hantering av krav som ska ge stöd för hela kravprocessen. Användarna kan fånga och hantera krav samt analysera dem. Verktyget blir ett spårbarhetscenter för hela projektets livscykel. Verktyget minskar omarbetning och utvecklingstiden samt ökar effektiviteten och verkansgraden, bland annat genom att stödja återanvändning av element från tidigare framgångsrika projekt (Visure, 2012).

*PACE* (Viewset) är ett webbaserat verktyg som stödjer identifiering, analys, specificering, modellering, verifiering och hantering. PACE kan identifiera enskilda krav, tabeller och bilder från Word-dokument och införliva dessa i sitt system. Ett mål med PACE är att underlätta samarbete och kommunikation genom bättre delning och tillgång till krav och förändringar. Systemet håller redan på vem som ändrat i krav, samt möjliggör spårbarhet från kundernas första krav till den slutliga specifikationen. Detta gör även att ej uppfyllda krav går att identifiera. PACE har även stöd för att skapa en informationsmodell över relevanta begrepp som berörs av kraven (ViewSet, 2011).

*ReqMan* är ett traditionellt kravhanteringsverktyg, där fokus ligger på att samla kraven i en databas, samtidigt som kraven kan presenteras och editeras som ett vanligt dokument (RequirementOne Inc., 2008; RequirementOne Inc., 2012).

*inteGREAT* är ett verktyg som tillåter användare att definiera, analysera, simulera och dokumentera krav automatiskt för alla olika typer av roller. Fördelarna med detta är att det snabbar upp intressentarbetet avsevärt i en av de mest kritiska och tidskrävande faserna av kravhantering. Ytterliggare fördelar med verktyget anses vara dess förmåga att automatisk generera kravdokument samt ett trettiotal olika typer av diagram. Verktyget stödjer identifiering-, analys-, validering- samt förvaltningsfaserna inom kravhantering (eDevTech, 2011).

# 5 Diskussion

Kravhantering är den enskilt viktigaste aktiviteten för att lyckas med systemutveckling (Young, 2001). Men dess genomförande är långt ifrån trivialt, och bristfälliga kravspecifikationer leder bland annat till felaktiga system samt onödiga kostnader för drift och underhåll. Dessutom förloras den nytta för verksamheten som systemet skulle kunna bidra med. Syftet med denna studie var att studera tillgängliga verktyg för kravhantering. Studien baserades på en litteraturstudie och en översiktstudie, vilka analyserar 34 vetenskapliga artiklar och 15 kommersiella verktyg.

Den kvantitativa sammanställningen av materialet i Tabell 1 visar att verktygen identifierade i litteraturstudien till största delen gav stöd till aktiviteten specificering. Den kvantitativa sammanställningen av de kommersiella verktygen visar att det var få verktyg som stödde kravarbete inom aktiviteten specificering och generellt stödde de kommersiella verktygen fler faser i kravhanteringsprocessen jämfört med de vetenskapliga som i högre grad begränsades till en aktivitet.

Avseende identifiering av krav pekade två vetenskapliga artiklar (AlAli & Issa, 2011; Feja et al., 2011) och det kommersiella verktyget (*IRqA*) på att återanvända kunskap från tidigare systemutveckling. Genom att spara tidigare processer, modeller, användningsfall och formella krav i databaser och strukturerat tillgängliggöra dessa för pågående utvecklingsprojektet ges möjlighet till kvalitetshöjning avseende att lyckas identifiera en mer fullständig uppsättning krav från användarna. För de kommersiella verktygen återfinns bland annat *GMARC* som ger stöd för identifiering av en komplett uppsättning krav genom att ställa strukturerade frågor direkt till användare. De kommersiella verktygen kan ses vara i högre grad utformade för att stödja hantverket vid identifiering av krav i direkt kontakt med användare (*Acclaro DFSS*) eller dokument (*PACE, Polarion Requirments management).*

Den största andel av de vetenskapliga artiklarna berörde verktyg som stödjer specificering av krav, till skillnad mot de kommersiella verktygen där specificering var den aktivitet där minst fokus för verktygen hittades. I de flesta fall specificeras krav i naturligt språk, vilket är en nackdel då dess inneboende tvetydighet, både semantiskt och syntaktiskt, gör att olika parter i systemutvecklingen kan tolka kraven på olika sätt. Analys och bearbetning av krav skrivna i naturligt språk utgör resurskrävande hantverk med stor risk för felaktigheter. Bland de vetenskapliga artiklarna identifierades verktyget *QUARS* (Bucchiarone et al., 2005; Lami et al., 2005) vilket hjälper kravanalytikern att i efterhand peka ut svagheter i kravspecifikationen, medan verktyg *HeRA* (Knauss et al., 2009) hjälper till att korrigera svagheter redan under författandet av kravspecifikationen. För att undvika de tvetydigheter som uppkommer i samband med att krav uttrycks i naturligt språk kan istället krav uttryckas med formella

språk. Formella språk har en strikt definierad terminologi och en tydlig syntax, vilket innebär att språken kan tolkas av datorer och därmed nyttjas vid automatisk bearbetning av krav. Automatisk bearbetningen ses eftersträvansvärt då en mindre resursåtgång för analys och bearbetning av krav kan uppnås. De kommersiella verktygen *ReqMan* och *Polarion Requirements management* stödjer bearbetning genom att samla krav i en databas och presentera dem som ett sammanhängande dokument. I kontrast till detta finns *inteGREAT* och *CASE spec* där dokument automatiskt genereras utifrån de identifierade kraven genom filtrering, gruppering och sortering. En av nackdelarna med formella språk är att de som hanterar krav måste ha en adekvat utbildning för att hantera språket. För att underlätta för de som hanterar krav i formella språk finns exempelvis Siy et als (2007) prototyp som transformerar krav i naturligt språk till formella språk för att sedan bearbeta kraven. Några av de i litteraturstudien beskrivna verktygen stödjer autonom rättning av tvetydigheter vid översättning av krav till formellt språk där kraven sedan kan bearbetas. Exempel på dessa verktyg är rapporterade av Weston et al. (2009); Popescu et al. (2008) och Umber et al. (2011). De exempel som beskrivs i litteraturstudien är dessvärre testade på små mängder information under laborativa förhållanden och enbart på ett naturligt språk, engelska.

Validering av faktiska system kan inte göras förrän sent i systemutvecklingen och då med tillgång till den faktiska miljön samt de avsedda användarna och uppgifterna. För att upptäcka felaktigheter hos system tidigt i utvecklingen påvisas i litteraturen hur olika valideringar av krav baserade på modeller, aktivitetsdiagram, testfall och prototyper kan utföras med varierande grad av automatisering (Robinson-Malett et al., 2010; Matos & Sousa, 2010; Ogata & Matsuura, 2010). De kommersiella verktygens stöd för kravvalidering syftade bland annat till att identifiera ej uppfyllda krav (*PACE*), ge en översikt av krav som ännu inte validerats (*Psoda*) och verifiera att identifierade krav tagits omhand i slutprodukten (*Acclaro DFSS*).

Det största tillämpningsområdet för de kommersiella verktygen var förvaltning av krav och de flesta verktygen hanterar bibehållandet av spårbarheten hos kraven. Andra exempel på stöd är att länka samman användningsfall med testfall (*CASE spec* och *Polarion Requirements Management*), organisera kraven hierarkiskt (*CASE spec* och *MKS integrity*), utföra konsekvensanalyser (*Reqtify*), skapa informationsmodeller över begrepp (*PACE*) eller hålla reda på vem som får/kan/har ändrat ett krav och varför (*Polarion Requirements Management, TraceCloud* och *PACE*). De vetenskapliga artiklarna fokuserade till stor del istället på systemets hela utveckling (Monteiro et al., 2009; Arpinen et al., 2011) och spårbarhet (Hong et al., 2010; Deshmuk & Wadhwa, 2007).

## 5.1 Slutsatser

Den genomförda litteraturstudien påvisar att automatiserad kvalitetssäkring inte är ett ämne som framstår som särskilt prioriterat inom kravhanteringsområdet. Flertalet av de verktyg som beskrivs är inte heller direkt avsedda för kvalitetssäkring, utan fokuserar snarare på att ge stöd för att genomföra aktiviteterna än för att säkerställa aktiviteternas kvalitet. Med tanke på den stora informationsmängd som måste hanteras inom kravhantering är det emellertid högst troligt att de aktiviteter som stödjs av automatiserade verktyg erhåller högre kvalitet än de aktiviteter som inte utförs med något verktygsstöd.

Litteraturstudien påvisar även att långt från alla verktyg i nuläget automatiserar kravhanteringsprocessen och en skillnad kan även ses gällande automatiseringsnivån för verktygen. Flertalet av de verktyg som utvecklas inom forskarvärden är relativt smala verktyg som ännu befinner sig på en konceptuell- eller prototypnivå. Detta står i kontrast till de kommersiella verktygen som ofta är bredare med fler funktioner och stödjer kravhanteringen på ett mer omfattande sätt. Bland de kommersiella verktygen återfinns flertalet databasbaserade helhetslösningar som även sträcker sig delvis utanför kravhanteringen och även utgör ett stöd för projektledning. Bland alla dessa verktyg är den vanligaste förekommande lösningen baserat på databaser som stödjer spårbarhet, strukturering och återanvändning av krav.

De vetenskapliga artiklarna visade att den största delen av det senaste arbetet avseende kvalitetssäkring av krav utförs inom området specificering. Flera av det verktyg som utvecklas inom området berör hanteringen av modeller, men även verktyg för att hantera naturligt språk och kravspecifikationer påträffas. Däremot har verktyg som berör områden som identifiering och analys av krav varit relativt få.

En välgrundad och välskriven kravspecifikation ger förutsättningar att lyckas med systemutveckling. Detta kräver dock en betydande arbetsinsats, inte minst när det gäller att kvalitetssäkra krav. Att inspektera och granska kravspecifikationer är tidskrävandet och relativt enahanda arbete. Detta medför att det finns skäl att vidare studera möjligheterna att skapa verktyg som stödjer och automatiserar detta.

# 6 Referenser

3SL (2012) *Cradle Modules Overview*. http://www.threesl.com/pages/Cradle/English/Content/Products/overview.php [2012-03-13]

AlAli, A. I., & Issa, A. A. (2011). Towards well documented and designed agile software development. *Proceedings of World Academy of Science, Engineering and Technology*, 73, pp. 126-131.

Alexander, I. (2012). *Requirements Tools*. http://easyweb.easynet.co.uk/~iany/other/vendors.htm [2012-03-14]

Alrajeh, D., Ray, O., Russo, A., & Uchitel, S. (2007). Extracting requirements from scenarios with ILP. In Muggleton S., Otero R. & Tamaddoni-Nezhad A.(Eds.), *Inductive Logic Programming, 16th International Conference, ILP 2006, Santiago de Compostela, Spain, August 24-27, 2006, Revised Selected Papers* (pp.64-78). Springer Berlin / Heidelberg.

Arpinen, T., Hämäläinen, T. D., & Hännikäinen, M. (2011). Meta-model and UML profile for requirements management of software and embedded systems. *Eurasip Journal on Embedded Systems*, 2011.

Autili, M., & Pelliccione, P. (2008). Towards a graphical tool for refining user to system requirements. *Electronic Notes in Theoretical Computer Science*, 211(C), pp. 147-157.

Axiomatic Design Solutions, Inc. (2006) *Acclaro DFSS Summary*. http://www.dfss-software.com/dfss_summary.asp [2012-03-13]

Beck, K. (2001). *Extreme programming explained: Embrace change*. Addison-Wesley.

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

Bucchiarone, A., Gnesi, S., & Pierini, P. (2005). Quality analysis of NL requirements: An industrial case study. *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 390-394.

Cao, L. & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Software*, 25(1), pp. 60–67.

Carrillo de Gea, J.M., Nicolás, J., Alemán, J.L.F., Toval, A., Ebert, C. &Vizcaíno, A. (2011). Requirements Engineering Tools, *Software, IEEE* , vol.28, no.4, pp.86-91.

Cognition Corporation (2012). *Cognition Cockpit - Requirements Management Software*. http://www.cognition.us/cockpit_overview.html [2012-03-14]

Computer System Architects Ltd. (2002). *GMARC*. http://www.informeng.com/Documents/ [2012-03-13]

Dascalu, S., Fritzinger, E., Debnath, N., & Akinwale, O. (2006). STORM: Software tool for the organization of requirements modeling. *2006 IEEE International Conference on Electro Information Technology*, pp. 250-255.

Matos, E. C. B., & Sousa, T. C. (2010). From formal requirements to automated web testing and prototyping. *Innovations in Systems and Software Engineering*, 6(1), pp. 163-169.

Deshmukh, N., & Wadhwa, S. (2007). A meta model for iterative development of requirements leveraging dynamically associated prototyping and specification artifacts. *Proceedings - 15th IEEE International Requirements Engineering Conference, RE 2007*, pp. 343-349.

eDevTech (2011). *inteGREAT Requirement Studio*. http://www.edevtech.com/requirements_studio.html [2012-03-13]

Eklöf, M., Hallberg, N., Hansson, J., Sjödin, L. & Sparf, M. (2009). *Behovsanalys avseende Försvarsmaktens förbandsmålsättningsarbete*, (FOI-R--2917--SE), Linköping: Totalförsvarets forskningsinstitut (FOI).

Feja, S., Witt, S., & Speck, A. (2011). BAM: A requirements validation and verification framework for business process models. *Proceedings - International Conference on Quality Software*, pp. 186-191.

Firesmith, D. (2003). Specifying Good Requirements, *Journal of Object Technology*, 2(4): 77-87.

Garbers, B., & Periyasamy, K. (2006). A light-weight tool for teaching the development and evaluation of requirements documents. *Proceedings of the Annual ASEE Conference and Exposition, Washington, DC: American Society for Engineering Education, 2006*.

Georgiades, M. G., & Andreou, A. S. (2010). Automatic generation of a software requirements specification (SRS) document. *Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, ISDA'10*, pp. 1095-1100.

Ghazel, M. (2011). Using graph-based techniques for temporal requirements engineering. *2011 IEEE GCC Conference and Exhibition, GCC 2011*, pp. 124-127.

Goda Software (2012). *CASE Spec*. http://www.analysttool.com/CASESpec/product-specifications/ [2012-03-13]

Greensoft (2010). *Reqtify*. http://www.geensoft.com/en/article/reqtify [2012-03-14]

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

Hallberg, N., Lewau, N., Hansson, J., Granlund, H., Nilsson, S., Harladsson, J., & Karlzén, H. (2011). *Kvalitetsbaserad ledningssystemutveckling: Metoder och principer*, (FOI-R--3358--SE), Linköping: Totalförsvarets forskningsinstitut (FOI).

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

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

ISO/IEC/IEEE 24765:2010 Systems and software engineering — Vocabulary. Geneva, Switzerland: International Organization for Standardization.

Kamalrudin, M., Hosking, J., & Grundy, J. (2011). Improving requirements quality using essential use case interaction patterns. *Proceedings - International Conference on Software Engineering*, pp. 531-540.

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

Kulak, D. & Guiney, E. (2003). *Use Cases: Requirements in context.* Upper Saddle River, NJ: Addison-Wesley.

Lami, G., Gnesi, S., Fabbrini, F., Fusani, M., & Trentanni, G. (2005). An automatic tool for the analysis of natural language requirements. *Computer Systems Science and Engineering*, 20(1), pp. 53-62.

Larman, C. (2004). *Agile & Iterative Development. A Manager´s Guide.* Addison-Wesley.

He, Q., & Antón, A. I. (2009). Requirements-based access control analysis and policy specification (ReCAPS). *Information and Software Technology*, 51(6), pp. 993-1009.

Hong, Y., Kim, M., & Lee, S. (2010). Requirements management tool with evolving traceability for heterogeneous artifacts in the entire life cycle. *8th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2010*, pp. 248-255.

Jani, H. M., & Mostafa, S. A. (2011). Implementing case-based reasoning technique to software requirements specifications quality analysis. *International Journal of Advancements in Computing Technology*, 3(1), pp. 23-31.

Knauss, E., Schneider, K., & Stapel, K. (2009). Learning to write better requirements through heuristic critiques. *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 387-388.

Knieke, C., Huhn, M., & Lochau, M. (2008). Modeling and validation of executable requirements using live activity diagrams. *Proceedings - 6th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2008*, pp. 51-58.

Koo, S. R., Seong, P. H., Yoo, J., Cha, S. D., & Yoo, Y. J. (2005). An effective technique for the software requirements analysis of NPP safety-critical systems, based on software inspection, requirements traceability, and formal specification. *Reliability Engineering and System Safety*, 89(3), pp. 248-260.

Mavin, A., Wilkinson, P., Harwood, A., & Novak, M. (2009). *EARS (Easy Approach to Requirements Syntax)*. 17th IEEE International Requirements Engineering Conference, pp. 317-322.

Monteiro, M. R., Ebert, C., & Recknagel, M. (2009). Improving the exchange of requirements and specifications between business partners. *Proceedings of the IEEE International Conference on Requirements Engineering*, pp. 253-260.

Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (ICSE '00). ACM, New York, NY, USA, 35-46.

Ogata, S., & Matsuura, S. (2010). Evaluation of a use-case-driven requirements analysis tool employing web UI prototype generation. *WSEAS Transactions on Information Science and Applications*, 7(2), pp. 273-282.

Orcanos (2009) *QPack Requirements Tool for Requirements Management*. http://orcanos.com/Requirements_management.htm [2012-03-13]

Patel, R., & Davidsson, B. (2003) *Forskningsmetodikensgrunder*. Lund: Studentlitteratur.

Polarion Software (2012). *Polarion Requirements*. http://www.polarion.com/products/requirements/index.php [2012-03-13]

Popescu, D., Rugaber, S., Medvidovic, N., & Berry, D. M. (2008). Reducing ambiguities in requirements specifications via automatically created object-oriented models. *Lecture Notes in Computer Science*, 5320, pp. 103-124.

Psoda (2011). *Psoda*. http://www.psoda.com/cms.php/home [2012-03-13]

RequirementOne Inc. (2008). *RequirementOne ReqMan Getting Started Guide v1.1*. http://www.scribd.com/doc/4028627/RequirementOne-ReqMan-Getting-Started-Guide [2012-03-13]

RequirementOne Inc. (2012). *RequirementOne*. http://www.requirementone.com/Project-Management-Platform [2012-03-13]

Robinson-Mallett, C., Grochtmann, M., Köhnlein, J., Wegener, J., & Kühn, S. (2010). Modelling requirements to support testing of product lines. *ICSTW 2010 - 3rd International Conference on Software Testing, Verification, and Validation Workshops*, pp. 11-18.

Romero-Mariona, J., Ziv, H., & Richardson, D. (2010). ASSURE: Automated support for secure and usable requirements engineering. *ISSTA'10 - Proceedings of the 2010 International Symposium on Software Testing and Analysis*, pp. 279-282.

Rubin, E. & Rubin, H. (2011). Supporting Agile software development through active documentation. *Requirements Engineering*, 16(2), pp. 117-132.

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

Siqueira, F. L., & Silva, P. S. M. (2011). Transforming an enterprise model into a use case model using existing heuristics. *2011 Model-Driven Requirements Engineering Workshop, MoDRE 2011*, pp. 21-30.

Siy, H., Aryal, P., Winter, V., & Zand, M. (2007). Aspectual support for specifying requirements in software product lines. *Proceedings - International Conference on Software Engineering*.

Sommerville, I. & Sawyer, P. (1997) Requirements engineering: a good practice guide, Chichester, Wiley.

Söderström, J. (2010). *Jävla skitsystem!: Hur en usel digital arbetsmiljö stressar oss på jobbet - och hur vi kan ta tillbaka kontrollen.* [Stockholm: Publit Sweden].

The Product Development Company (2012). *Integrity - Accelerating Innovation in Software Intensive Products*. http://www.mks.com/platform/our-product [2012-03-14]

TraceCloud (2010). *TraceCloud*. http://www.tracecloud.com [2012-03-14]

Udomchaiporn, A., Prompoon, N., & Kanongchaiyos, P. (2006). Software requirements retrieval using use case terms and structure similarity computation. Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 113-120.

Umber, A., Bajwa, I. S., & Asif Naeem, M. (2011). NL-based automated software requirements elicitation and specification. *Communications in Computer and Information Science*, 191(2), pp. 30-39.

Valderas, P., & Pelechano, V. (2007). Improving communication in requirements engineering activities for web applications. *Lecture Notes in Computer Science* (4607), pp. 242-247.

Videira, C., Carmo, J. L., & Silva, A.R. (2005). The ProjectIT-RSL language overview. *Lecture Notes in Computer Science*, 3297, pp. 269-272.

ViewSet (2011). *Introducing PACE*. http://www.viewset.com/index.php/products-pace-overview [2012-03-13]

Visure (2012). *IRQA*. http://www.visuresolutions.com/irqa-requirements-tool [2012-03-14]

Weston, N., Chitchyan, R., & Rashid, A. (2009). Formal semantic conflict detection in aspect-oriented requirements. *Requirements Engineering*, 14(4), pp. 247-268.

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

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

Zachos, K., Maiden, N., & Howells-Morris, R. (2008). Discovering web services to improve requirements specifications: Does it help? *Lecture Notes in Computer Science*, 5025, pp.168-182.