Kravhantering

Introduktion

”Som man frågar får man svar”. Detta ordspråk kan med fördel kan användas på området kravspecificering. En bra kravbild är bland de mest avgörande framgångsfaktorerna för ett systemutvecklingsprojekt. En arkitektur är ett delmål i detta arbete. Därför har kraven stor påverkan på hur arkitekturen utformas [Lawrence].

Det är ett välkänt faktum att det är svårt både att finna kraven och att formulera dem på ett entydigt och begripligt sätt. Det räcker emellertid inte med att finna och formulera kraven, de måste även prioriteras, spåras och ändringshanteras. Slutligen behöver man validera att det resulterande systemet uppfyller kraven. Detta fordrar samverkan mellan flera intressenter under hela systemets livscykel. Nästa avsnitt ger en sammanfattning av vad som bör ingå i området kravhantering.

Vad är ett krav?

Mål och krav

Målen är det resultat i verksamheten som en beställare önskar uppnå med en önskad produkt eller tjänst från en leverantör. Målen är ett uttryck för beställarens behov. Ur målen härleder beställaren (ev i samverkan med leverantören) kraven på produkten/tjänsten. Kraven är en förädling av målen och anger vad beställaren vill att leverantören skall uppfylla med sin leverans.

Målen uttrycks bäst i verksamhetstermer medan kraven bäst formuleras i termer som så nära som möjligt ansluter till produktens/tjänstens egenskaper. Produktnära mål och krav kan uttryckas i olika dimensioner. Vanligen skiljer man på krav som rör funktionalitet och krav som rör övriga egenskaper. Funktionaliteten anknyter till uppgifter inom verksamheten. Kategorin övriga egenskaper behöver preciseras ytterligare. Detta kan ske på många olika sätt som kommer att framgå av följande avsnitt.

Uppfattningen om vad som menas med ett krav är många gånger diffus. Tolkningarna varierar från vaga önskemål till mycket tydliga och mätbara preciseringar. Här följer en definition hämtad från VI rapporten *Framgångsrik kravhantering* [VI 98].

> Ett krav är en önskvärd egenskap eller funktion hos ett IT-system.

Denna definition betraktar som synes krav som något som kan uppfyllas helt, delvis eller inte alls. Ett krav är alltså inte begränsat till något som ovillkorligen måste vara uppfyllt. Med denna tolkning måste man ägna mer uppmärksamhet åt hur man bedömer graden av kravuppfyllnad. Här följer en kompletterande definition på krav som kopplar dess realisering till hur det är formulerat.

> Ett krav skall vara formulerat så att det är möjligt att avgöra i vilken grad man har uppnått kravet i den slutliga produkten.

Av detta följer att det är viktigt att kunna gradera och mäta hur väl ett krav uppfylls. En betydelsefull egenskap hos ett krav är alltså att det skall vara mätbart.

Ett krav har alltid ett ursprung, ett motiv och ett realiseringsobjekt. Ursprunget är en intressent, som har ett eller flera behov, som det aktuella systemet bör tillfredsställa. Realiseringsobjektet är en del (t ex en komponent eller en egenskap) av det resulterande IT-systemet.

Figur 1. Sambandet intressent, krav och system.

Funktionskrav och egenskapskrav

Vanligen skiljer man på krav som rör produktens/systemets funktionssätt och krav som rör dess mer kvalitativa egenskaper som t ex tillförlitlighet och säkerhet. Dessa senare *egenskapskrav* brukar ibland sammanfattas under rubriken ”icke funktionella krav” eller ”kvalitativa krav”. I engelskspråkig litteratur används även benämningen ”attribute requirements” för egenskapskraven.

Vilka krav bör man ställa på ett krav?

En vanlig men tyvärr alltför idealistisk bild av en kravspecifikation (och därmed av de enskilda kraven) är att den skall vara fullständig, korrekt och entydig. Även om detta i praktiken är mycket svårt att uppnå är det ett mål värt att sträva efter.

Capers Jones [Jones 94, sid 93] har genom analys av sina mångåriga mätningar på programvaruprojekt kommit fram till att kraven på ett normalstort system (storleken några hundra tusen rader kod) ändras med en takt av ca 1% per månad. Av detta kan man sluta sig till att det är en omöjlig uppgift att bygga ett system som svarar mot kraven så som de formulerades när projektet startade om inte systemet är mycket litet. Detta är en av anledningarna till att man numera förordar att ett system byggs inkrementellt, dvs med delleveranser som stegvis utökar det funktionella innehållet.

Egenskapskraven är vanligen mer stabila över tiden än de funktionella kraven och dessutom mindre känsliga för smärre förändringar i den verksamhet som IT-systemet skall stödja. Här finns alltså bättre förutsättningar att tidigt klara ut krav som är gränssättande när det gäller teknikval.

Trots dessa svårigheter är det rimligt att sträva efter att formulera kraven entydigt och begripligt. VI-rapporten Framgångsrik kravhantering [VI 98] formulerar 24 karakteristika för hur en

uppsättning krav bör formuleras och hanteras. De omfattar både formulering (entydig, komplett, korrekt, begriplig, etc.) och hantering (spårbarhet, prioritering, versionshantering, etc.).

Kravhanteringsprocessen

De aktiviteter som tillsammans fordras för att åstadkomma en bra kravhantering bildar en kravhanteringsprocess. De är:

- Insamling
- Dokumentation
- Prioritering
- Verifiering och validering
- Förvaltning

Uppräkningen ovan innebär inte att aktiviteterna måste genomförs i denna ordning även om det finns ett visst logisk beroende mellan dem. Det går t ex inte att dokumentera ett krav innan det är identifierat etc.

De typer av intressenter, som samverkar i denna process, är *leverantörer* (försäljare, analytiker, arkitekter, systemutvecklare) och *kunder* (beställare, användare). Leverantören önskar få klarhet i vad som skall konstrueras medan kunden vill ha ett IT-system som stöder verksamheten. Detta leder till olika fokus i de olika aktiviteterna. En svårighet i kravarbetet är att överbrygga dessa skilda mål och skapa en gemensam syn på det tilltänkta IT-systemet. En viss hjälp kan man få från standarder av typen IEEE 1471 Recommended Practice for Architectural Description of Software-Intensive Systems,som rekommenderar användning av ett antal perspektiv som kopplas till kraven från respektive intressent.

Kravarbetet kan utföras av endera av dessa intressenter. Med tanke på deras olika fokus kommer då den resulterande kravspecifikationen att betona helt olika saker. Detta kan leda till problem i det följande utvecklingsarbetet. En samverkan mellan olika intressenter ger de bästa förutsättningarna för att uppnå ett balanserat resultat. Detta kan vara svårt att åstadkomma i en upphandlingssituation där ju en potentiell leverantör inte bör specificera kraven på det system som han sedan i konkurrens med andra skall offerera att utveckla. Ett sätt att hantera detta är att inte tillåta dem som medverkar i kravarbetet att medverka i utvecklingsarbetet.

Under de olika aktiviteterna kan olika metoder behöva tillämpas. Det finns även metoder som spänner över flera aktiviteter. Ett exempel är Quality Function Deployment (QFD). Denna metod har sitt ursprung i japansk bilindustri och fokuserar på att finna och prioritera kundkraven med hänsyn till ett antal bivillkor. För att få alla perspektiv belysta arbetar man i grupper sammansatta av personer från olika kompetens- och intresseområden. Som stöd i bedömningen av kraven använder man det sk kvalitetshuset, som är en flerdimensionell matris som relaterar kraven till varandra.

Insamling

Funktionskrav

Dessa krav kan vara ett resultat av en verksamhetsanalys som visat på områden där IT kan ge förbättringar. Kraven är liktydiga med de funktioner som måste finnas för att systemet skall ge stöd för en användare när det gäller att lösa dennes arbetsuppgifter. För ett delsystem, t ex komponent i ett större system eller ett stödjande system, utgör kraven de funktioner, gärna beskrivna i form av gränssnitt, som fodras för att det omgivande systemet skall kunna utföra sina uppgifter.

Ett effektivt sätt att härleda funktionskrav är därför att utgå från de enskilda användarna (gärna generaliserade i form av roller, aktörer) och beskriva de enskilda uppgifter (användningsfall) som varje aktör har att utföra. Ett omgivande system kan också ses som en aktör. Detta är schematiskt just det arbetssätt som utvecklingsmetoden RUP (Rational Unified Process) förordar. Liknande arbetssätt har med framgång använts tidigare under rubriken scenariobeskrivningar eller rutinskisser. De senare används ofta i samband med beskrivningar av administrativt inriktade verksamheter och omfattar då även manuella rutiner. Användningsfallen fokuserar på gränssnittet mellan aktören och IT-systemet, dvs hur IT-systemet stöder aktören i att lösa en given uppgift. Ett arbetssätt vid administrativ systemutveckling kan därför vara att starta med rutinbeskrivningar och bedöma vilka uppgifter som lämpligen kan och bör stödjas av ett IT-system. Dessa uppgifter är sedan underlag för användningsfall.

De frågor som ger svar på de funktionella kraven är alltså VEM och VAD.

Exempel på andra sätt att finna funktionskrav är intervjuer, observationer, brainstorming.

Ett problem i kravarbetet är att finna en lämplig nivå på kraven. Detta gäller framför allt funktionskraven. Om ett krav detaljeras för långt kan det komma att föreskriva hur en funktion skall realiseras, vilket begränsar utrymmet för konstruktören. Kravet övergår till att bli en konstruktionsbeskrivning. En sätt att undvika detta kan vara att dela in det aktuella systemet i delsystem eller komponenter. Kraven kan då begränsas till den funktionalitet som respektive komponent skall kunna leverera (jfr begreppet tjänstebaserad arkitektur). Men då föregriper man arkitekturen (vilket kan vara helt rimligt i vissa fall)

Egenskapskrav

Denna typ av krav är betydligt mer svårfångade än funktionskraven och fordrar en analys av verksamheten ur andra perspektiv. Genom att formulera frågor av typen ”vad händer om” kan man exempelvis få en bättre bild av kraven på tillförlitlighet och tillgänglighet. Det går inte att formulera en enkel metod för att finna dessa krav. Eftersom kravarbetet lätt fokuseras på funktionskraven gäller det att kunna avgöra vilka kvaliteter som systemet behöver för att kunna leverera dessa funktioner. Detta kan underlättas om man försöker klara ut under vilka omständigheter systemet kommer att användas. Frågor som kan underlätta kravformuleringen är:

HUR MYCKET, NÄR, VAR, TILL VEM, VILKET PRIS, osv.

Dessa frågor visar på betydelsen av att kunna gradera kraven i något mått. Se mer om detta under dokumentation.

Att kategorisera krav

En god inledande distinktion av olika typer av krav är att skilja mellan funktionskrav och egenskapskrav. Kategorin egenskapskrav rymmer emellertid så många aspekter att det är lämpligt att göra en ytterligare kategorisering för att göra kravbilden mer tillgänglig. En bra grund för en sådan indelning ges i standarden *ISO 9126 Software Product Evaluation* som beskriver hur man kan strukturera framför allt de egenskapskraven. Denna standard från 1991 är välgörande kortfattad (13 sidor) och anvisar även en grov utvärderingsprocess.

ISO 9126 anger att en produkts kvalitet skall bedömas efter följande sex huvudrubriker. Ett appendix i standarden innehåller en modell för hur dessa kan tolkas genom att komplettera med ett antal underavdelningar per rubrik (i fri översättning):

1. Funktionalitet: lämplighet, korrekthet, samverkan, överensstämmelse med standard, säkerhet
2. Tillförlitlighet: mognad, feltolerans, återgång (efter fel), tillgänglighet
3. Användbarhet: begriplighet, inlärningskrav, handhavande
4. Effektivitet: tidsaspekter (t ex prestanda), resurskrav (t ex skalbarhet)
5. Underhållsbarhet: analyserbarhet, modifierbarhet, konfigurerbarhet, provningsbarhet
6. Flyttbarhet: anpassningsbarhet, installationskrav, överensstämmelse med standard, ersättningsbarhet

För samtliga dessa egenskaper finns en kort förklarande beskrivning. Det är uppenbart att det är denna typ av krav som är gränssättande för den grundstruktur hos ett system som vi kallar för arkitektur. Detta är anledningen till följande påstående:

> Det är egenskapskraven som styr utformningen av en systemarkitektur på alla dess nivåer.

Dokumentation

Ett krav behöver beskrivas så att det är dels är begripligt och dels går att avgöra om (hur väl) kravet är uppfyllt i det färdiga systemet. Dessutom bör de enskilda kraven finnas sammanställda så att det går att bilda sig en överblick över helheten. Standarden IEEE 830 beskriver vad ett kravdokument bör innehålla och hur det kan disponeras. Den sammanfattas nedan i avsnittet om kravspecifikationen.

Funktionskrav

De funktionella kraven är svåra att beskriva men enkla att gradera. De är i huvudsak binära. Antingen finns funktionen eller också så finns den inte. Om funktionen kopplas samman med effekter som man vill uppnå som ett resultat av funktionen så blir det däremot svårare. Exempelvis kravet ”systemet skall stödja ärendebevakning, som skall kunna utföras på ett rationellt sätt”. En lösning kan vara att dela upp detta krav i två separata krav, som dels beskriver vad funktionen innebär och dels preciserar vad som menas med rationellt arbetssätt.

Beskrivningar av funktionskrav kan ske i informellt som löpande text, styrt av mallar som t ex är vanligt för användningsfall, eller höggradigt formaliserat i något specifikationsspråk. Beskrivningar med diagram såsom klassdiagram, användningsfallsdiagram och sekvensdiagram i UML ger en viss formalisering men som behöver kompletteras med beskrivande text.

Egenskapskrav

Ett egenskapskrav behöver kunna kvantifieras på ett mer nyanserat sätt än ett funktionskrav. Under rubriken kravinsamling nämndes följande frågor som kan underlätta arbetet med att finna egenskapskraven:

> HUR MYCKET, NÄR, VAR, TILL VEM, VILKET PRIS, osv.

Det sätt som frågorna formulerats belyser vikten av att kunna gradera kraven i något mått. Tom Gilb [Gilb 77, Gilb 98] har föreslagit följande mall för denna typ av krav:

> *Krav-Id:* Ge ett kompakt namn eller välj en unik identifierare på kravet.
>
> *Beskrivning:* En kortfattad beskrivning av idén eller meningen med kravet.
>
> *Skala:* Vilket mått och vilken skala som används för att gradera kravnivån
>
> *Nivå:* Anger till vilken nivå som kravet skall uppfyllas. Här kan man tillfoga graderingar som Plan - i vilken takt, Måste - en ovillkorlig miniminivå, Önskemål - en ”bästa” nivå utöver minimum, osv.
>
> *Historik:* Visar vad man hittills uppnått och i vilken takt (inom verksamheten eller vad som är bästa möjliga nivå, dvs rekordet).

Utöver dessa rubriker kan man komplettera med beskrivningar av hur mätningar av kravuppfyllelsen skall gå till. Det är en bra försäkring mot framtida tolkningskonflikter om beställare och leverantör är överens om mått och mätmetod.

Kravspecifikationen

Kravspecifikationen är ett centralt dokument i all systemutveckling. Dess innehåll och utformning varierar beroende på typ av system och metod för kravinsamling. Tanken är att kravspecifikationen skall vara underlag för att inledningsvis träffa avtal mellan beställare och leverantör och senare vara styrande vid utveckling och förvaltning. Den bör därför inte innehålla

krav på projektutformning eller detaljer om systemets konstruktion annat än i form av bivillkor eller begränsningar. Ett (utvecklings)projekt existerar ju enbart under de inledande faserna av ett systems livscykel medan kraven gäller under systemets hela livslängd. Hur ett system utformas i detalj är upp till konstruktörerna att bedöma och bör inte ingå i kravspecifikationen eftersom det kan begränsa valmöjligheterna på ett olyckligt sätt. Vissa riktlinjer kan dock fordras, t ex anpassning till befintlig teknisk infrastruktur och får da tas upp under rubriker som begränsningar och bivillkor.

Med den indelning av processer för systemlivscykeln som rekommenderas av ISO 12207 Software Lifecycle Processes [ISO 12207] är kravspecifikationen ett resultat från den inledande huvudprocessen anskaffning (se detta avsnitt). Vi har emellertid här valt att sammanföra alla aktiviteter kring kravarbetet i en och samma process, kravhanteringsprocessen. Inte minst eftersom kraven förändras löpande ref Capers Jones

IEEE Std. 830 *Recommended Practice for Software Requirements Specification* [IEEE 830] innehåller en vägledning till hur en kravspecifikation kan disponeras. Den lämpar sig framför allt för tekniskt inriktade system men kan även användas för administrativa system. Dess främsta fördel är att den beskriver vad som bör ingå i en kravspecifikation och hur den kan struktureras. Som övergripande krav på en kravspecifikation anger standarden följande egenskaper som vi känner igen från tidigare som generella krav på krav: korrekt, entydig, komplett, konsistent, prioriterad, verifierbar, modifierbar och spårbar. Dessa begrepp förklaras och kompletteras med rekommendationer på lämpliga arbetsmetoder.

En kravspecifikation bör enligt IEEE 830 ha följande struktur:

1. Inledning
    a. Avsikt med specifikationen
    b. Typ av system
    c. Definitioner, förkortningar och referenser
    d. Översikt över den följande specifikationen
2. Översiktlig beskrivning
    a. Systemet/produkten i sin omgivning
    b. Funktionsöversikt
    c. Användarkaraktäristik
    d. Bivillkor och begränsningar
    e. Antaganden och beroenden
3. Kravförteckning
    a. Detaljerade krav organiserade enligt någon vald struktur

Kravförteckningen bör organiseras så att samhörande krav hålls ihop. Olika kravtyper bör få egna rubriker. Eftersom listan på funktionskrav vanligen är omfattande bör den indelas ytterligare. Exempelvis kan man strukturera användningsfall efter aktörer. För ett system som skall kunna lagra data i en databas fordras att dess struktur beskrivs under särskild rubrik. Gränssnitt mot andra system är ytterligare ett område som bör få en särskild rubrik i specifikationen.

Prioritering

Under utveckling och förvaltning av ett system uppkommer ofta situationer då man behöver kompromissa mellan motstridiga krav. Ofta är det fråga begränsningar i kalendertid, kostnad eller resurstillgång som gör att man måste begränsa funktionalitet eller ge avkall på något egenskapskrav. Det kan även gälla egenskapskrav som sinsemellan är motstridiga och där realisering av det ena kommer att ske på det andras bekostnad. Vi vet också att många krav kommer att ändras under utvecklingens gång. Hantering av dessa situationer underlättas avsevärt om kraven är prioriterade och inbördes rangordnade.

En grundläggande prioritering av funktionskrav är att indela dem i tre kategorier, tvingande (ofta kallade ”skallkrav”), önskvärda (”börkrav”) och trevligt att ha (??) (”??”).

Skallkraven är de funktioner eller egenskaper som är absolut nödvändiga för att systemet skall kunna fylla sin uppgift. De är inte förhandlingsbara. Är de inte uppfyllda är systemet värdelöst.

Börkraven är av den art att det går att klara sig utan motsvarande funktioner eller egenskaper men det kan leda till vissa besvär eller lägre effektivitet. De är med andra ord förhandlingsbara.

De kosmetiska kraven slutligen bidrar endast marginellt till systemets helhet.

Ett problem med många kravspecifikationer är att alltför många krav hamnat i kategorin skallkrav. Detta ger föga ledning vid projektplanering, t ex uppdelning i delleveranser (inkrement) eller när man behöver skära ned funktionaliteten av tids-, kostnads- eller resursskäl.

Krav inom samma kategori bör dessutom ha en inbördes rangordning. Vissa skallkrav kanske av olika skäl bör realiseras före andra. På samma sätt förhåller det sig med egenskapskraven. Där uppkommer ofta svårigheter genom att olika egenskaper kan stå i konflikt med varandra. Exempelvis kan för ett visst system kraven på prestanda och modifierbarhet vara satta så att de inte samtidigt realiseras till önskad nivå. Då fordras en rangordning som säger vilket av dem som i första hand skall tillfredsställas.

Vid inkrementell utveckling är det motiverat att realisera arkitekturdrivande användningsfall tidigt även om de inte ligger högst på kravlistan

Linköpingsföretaget Focal Point har tagit fram en metod för kravprioritering (www.focalpoint.se). Metoden bygger på parvis jämförelser mellan krav. Prioriteringen innebär att man placerar in kraven på en ordinalskala där ordningen säger att krav är mer eller mindre viktiga i förhållande till varandra men inte hur mycket. En förutsättning för jämförelserna är att man enats om efter vilka kriterier jämförelserna skall ske. Några uppenbara kriterier är kundvärde, tid och kostnad. Focal Point-metoden kan leda till optimeringar genom att den underlättar att finna krav som exempelvis ger högt kundvärde till låg kostnad eller andra kombinationer av urvalskriterier. Till metoden finns ett datorstöd som även har ett webbgränssnitt för distribuerad kravhantering.

Verifiering och Validering

Även med den bästa av arbetsgrupper kommer den resulterande specifikationen att var behäftad med fel och andra ofullkomligheter. Den behöver därför utsättas för någon form av granskning eller provning. Skilj på verifiering och validering som är kompletterande aktiviteter för att identifiera felaktigheter. Verifiering avser korrekthet (gör systemet saker på rätt sätt) och validering avser rimlighet (gör systemet rätt saker). Standarden ISO 12207 Software Lifecycle Processes placerar verifiering och validering i två separata stödjande processer.

Ett mycket resurseffektivt arbetssätt för bägge dessa aktiviteter är granskningar. Dessa kan ske individuellt eller i grupp. Målet är att en granskare skall gå igenom ett arbetsresultat, t ex ett dokument eller programkod i avsikt att finna ofullkomligheter. Ursprunget är från 70-talet då det tillämpades inom IBM (Fagan Inspections). Granskning av kravspecifikationer är något som absolut rekommenderas. En inledande svårighet är kan vara att finna och motivera lämpliga deltagare. Ekonomiskt och kvalitetsmässigt är det inte svårt att motivera granskningar. Det finns en uppsjö av studier som visar på effektiviteten. I samband med millennieskiftet gjorde tidskriften IEEE Software en rangordning av de mest betydelsefulla bidragen till programutvecklingsområdet det gånga seklet [McConnell]. Där kom granskning på första plats före sådana bidrag som användarmedverkan och OO.

Förvaltning

En kravspecifikation är inte ett statiskt dokument. Kraven kommer med säkerhet att ändras under systemets utveckling och senare stadier i livscykeln. En komplett kravprocess innehåller därför även en förvaltningsaktivitet. Denna omfattar framför allt ändringshantering.

Ändringshantering

För att avgöra omfattningen av ett ändrat eller tillkommande krav fordras att kraven organiserats så att det går att avgör hur de hänger ihop, vilka beroenden som finns mellan dem. Vid förvaltning av ett befintligt system är kravspecifikationen en bra utgångspunkt, särskilt om det går att avgöra hur kraven realiserats i systemet,. Det måste med andra ord finnas en spårbarhet från kraven till deras realisering.

Detaljer kring hur man kan organisera en ändringshantering framgår av avsnittet om systemförvaltning (ej med här).

Spårbarhet

Spårbarhet är principiellt enkelt men praktiskt svårt att åstadkomma. Det handlar dels om att kunna följa ett krav från dess ursprung till dess realisering och omvänt att kunna avgöra vilka krav som en programvaruenhet realiserar. Detta gäller framför allt funktionskrav. Egenskapskraven är ofta realiserade genom val av övergripande arkitektur och därför svåra eller tom omöjliga att spåra till vissa delar av systemet. Spårbarhet i kravspecifikationen innebär att det går att identifiera den intressent som är upphov till kravet.

Verktyg för kravhantering

En rationell hantering av kraven för stora och medelstora system underlättas avsevärt av någon form av verktygsstöd. Kraven kan t ex lagras i en databas. Detta underlättar sökningar och sammanställningar av krav enligt de kriterier och attribut som lagrats.

Några exempel på verktyg är *Requisite Pro* och *DOORS*. Bägge verktygen tillhandahåller en grovstruktur för kravhantering som går att anpassa och komplettera.

Referenser

[Davis] Davis A. *Software Requirements, Analysis and Specification*. Prentice Hall, 1990,

[Gilb 77] Gilb T . *Software Metrics*. Whintrop Publ. 1977.

[Gilb 97] Gilb T . *Advanced Software Requirements and Design Specification Workshop*. Unicom Seminars, April 1997

[IEEE 1471] *IEEE 1471 Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Std 1471-2000.

[IEEE 830] *IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specification*. IEEE 1993.

[ISO 9126] *ISO 9126 Software Product Evaluation*.

[Jones] Jones C. *Assessment and Control of Software Risks*. Prentice Hall. 1994.

[Lawrence] Lawrence, Wigers, Ebert. The Top Risks of Requirements Engineering. IEEE Software, Nov/Dec 2001, pp 62-63.

[McConnell] Steve McConnell, IEEE Software January/February 2000

[Thayer] Thayer R, Dorfman M, red. *Software Requirements Engineering*, 2nd ed. IEEE Computer Society Press, 1997.

[VI 95] Hjelte Mats, m fl. *Krav på krav*. VI rapport nr 2, 1995. Best nr V040038.

[VI 98] Karlsson Joachim. *Framgångsrik kravhantering, utgåva 2*. VI rapport nr 3, 1998. Best nr V040072.

[Weinberg] Gause D., Weinberg G. *Exploring Requirements, Quality before Design*. Dorset House, 1989.

Länkar

http://www.stickyminds.com Här finns en hel del bra material kring krav, QA och testning.