Kursplan

REQB®

Certifierad Kravhanterare

Grundnivå

Ver 2.0

2014

Software Quality Engineering Board (SQEB)





Grundtanke

Grundtanken med denna kursplan är att programvarukomplexiteten och vårt beroende av programvara fortsätter att öka. Resultatet gör att vi kräver att programvara, hårdvara och andra produkter har en hög nivå av felfrihet. Requirements Engineering Qualifications Board (REQB) har därför beslutat att skapa en enhetlig internationell standard inom området kravhantering. Standarder är som en gemensam terminologi, när du väl förstår termer kan du arbeta effektivt. För att kunna skapa ett sådant enhetligt språk i detta viktiga område som kravhantering är, så har förlagan till denna kursplan skapats av internationella experter samordnat av REQB.

Tillkännagivanden

Det engelska originalet (Syllabus, Certified Professional for Requirements Engineering) till denna kursplan har producerats av ett kärnteam från Requirements Engineering Qualifications Board Working Party Foundation Level (Edition 2013): Karolina Zmitrowicz (chair), Sergey Kalinov, Beata Karpinska, Andrey Konushin, Salvatore Reale, Alexander Lidner, Ingvar Nordström, Alain Ribault.

Bidragande till den svenska översättningen har varit Ninna Morin, Ingvar Nordström och Beata Karpinska.

Denna kursplan

Detta dokument baseras på REQB Certified Professional for Requirements Engineering utgivet av Requirements Engineering Qualifications Board, REQB i samarbete med Global Association for Software Quality, och har översatts till svenska av Software Quality Engineering Board, SQEB.

SQEB och REQB har kommit överens att följande villkor skall gälla:

1) Individer och kursarrangörer får använda denna kursplan som bas för kurser om referenser ges till REQB och SQEB som copyright-ägare av dokumentet och att all annonsering av kurser först kan omnämna kursplanen efter det att en officiell ackreditering av kursmaterialet har gjorts av SQEB (SQEB är av REQB erkänt som nationellt organ i Sverige).

2) Varje individ eller grupp av individer får använda denna kursplan som bas för artiklar, böcker eller andra skrifter om REQB och SQEB är nämnda som källan och copyright-ägare av kursplanen

Copyright © SQEB 2014

Innehållsförteckning

Inledning 5
1 Introduktion till krav 11
1.1 Definition av krav 13
1.1.1 Definition av problem, lösning och produkt 13
1.1.2 Definition och klassificering av krav 13
1.1.3 Vanliga kravattribut 15
1.1.4 Kravkvalitet 17
1.1.5 Kravhantering, kravledning och kravutveckling 19
1.2 Standarder och normer 21
2 Kravhanteringens sammanhang 23
2.1 Kravhantering i sitt sammanhang 24
2.2 Relaterade processer 26
3 Introduktion till kravhanteringsprocessen 27
3.1 Introduktion till kravhanteringsprocessen 28
3.2 Generisk kravhanteringsprocess 29
3.3 Roller och ansvarsområden 33
4 Kravledning 35
4.1 Introduktion till kravledning 37
4.2 Projekt- och riskhantering 38
4.3 Kravspårbarhet 41
4.4 Konfigurations- och ändringshantering 43
4.5 Kvalitetssäkring 46
5 Kravutveckling 49
5.1 Introduktion till kravutveckling 51
5.2 Kravinsamling 52
5.3 Kravanalys 62

© SQEB – Software Quality Engineering Board

5.3.1 Lösningsmodellering 65
5.4 Kravspecificering 71
5.5 Validering och verifiering av krav 75
6 Kravhantering i modeller 76
6.1 Utvecklings- och förvaltningsmodeller och angreppssätt 77
6.2 Mognadsmodeller 82
7 Verktygsstöd 84
7.1 Fördelar med verktyg 85
7.2 Verktygskategorier 86
7.3 Val av verktyg 87
8 Tabeller/Bilder 89
9 Standarder 90
10 Litteratur 91
11 Index 93

© SQEB – Software Quality Engineering Board

Inledning

Syfte med kursplanen

Denna kursplan definierar grundnivån i det program som syftar till att bli certifierad kravhanterare (REQB Certified Professional for Requirements Engineering, (CPRE)). Den är utvecklad av REQB i samarbete med Global Association for Software Quality (GASQ) och sedan översatt till svenska. REQB-programmet omfattar kravhanteringsprocessen för alla typer av IT-relaterade produkter, där en produkt kan bestå av programvara, hårdvara, tjänster och affärsprocesser, samt tillhörande dokumentation.

Kursplanen fungerar som en bas för kurshållare som ansöker om ackreditering som lärare. Alla avsnitt i kursplanen måste på motsvarande sätt ingå i kursmaterialet. Kursplanen kan också fungera som förberedelse för studenter inför en tentamen. Alla områden är därför föremål för en tentamen som kan tas antingen efter en genomgången kurs eller oberoende av kurs.

Examinering

Examinering för att bli certifierad kravhanterare på grundnivå (Foundation Level) är baserat på denna kursplan och ordlistan. Alla delar av kursplanen är därför med i examen. Examensfrågorna är inte nödvändigtvis knutna till enbart ett kapitel utan kan spänna över flera.

Examinationen består av flervalsfrågor (multiple choice).

Examinering kan göras efter deltagande i en ackrediterad kurs, i en öppen examinering (utan tidigare kurs) eller som deltagare i en examen som hållits efter en kurs.

Du finner detaljerad information rörande examineringstidpunkter på SQEB:s hemsida (www.sqeb.se), REQB:s hemsida (www.reqb.org) eller på GASQ:s hemsida (www.gasq.org).

Ackreditering

Kurshållare för kurs för REQB Certifierad Kravhanterare måste vara ackrediterad av Global Association for Software Quality eller SQEB. När det gäller kurs på svenska ger detta genom SQEBs försorg. Kurshållarnas material granskas för att kontrollera täckning och överensstämmelse med innehåll och inlärningsmål i kursplanen. En ackrediterad kurs överensstämmer med kursplanen. I slutet av en sådan kurs kan en officiell tentamen för Certified Professional for Requirements Engineering (CPRE-tentamen) äga rum.



Ackrediterade kurshållare kan kännas igen genom den officiella loggan för REQB Accredited Training Provider.

Internationellt

© SQEB – Software Quality Engineering Board

Denna kursplan har utvecklats i samarbete mellan internationella experter. Innehållet i denna kursplan kan därför ses som en internationell standard. Kursplanen gör det möjligt att utbilda och examinera internationellt på samma nivå.

Affärsfördelar

Mål, fördelar och huvudfokus för REQB Certified Professional för Requirements Engineering Foundation Level program presenteras i nedanstående tabell (Tabell 1 – Mål med programmet på grundnivå, dess fördelar och huvudfokus).

Mål	
Skaffa nya kvalifikationer	Alla programvaru-, hårdvaru- eller tjänstelösningar baseras på intressenters krav och strävar efter att tillfredsställa specifika affärsbehov. Bra kravhantering är nödvändig för att kunna leverera en lösning som överensstämmer med målgruppens behov. Att hantera kraven och utveckla dem till lösningsdesign är väldigt viktigt, men för att uppnå full framgång och leverera den bästa lösningen ska även kvalitet tas med i beaktande. Alla dessa aspekter täcks i Certifierad kravhanterare - grundnivå.
Öka kundnöjdheten	Kundnöjdhet uppnås när kunden upplever att lösningen möter deras förväntningar och behov. Förbättrad kravhantering minskar skillnader mellan förväntningar och upplevelse av konformitet. Högre kvalitet hos den slutliga produkten stärker också kundens lojalitet.
Minska kostnader för utveckling och uppföljning	Bra kravhantering minskar projekt- och produktrisker och bidrar till att undvika kostnader kopplat till omarbete som följd av, till exempel, avvikelsen mellan kundens förväntningar och effektiviteten hos den lösning som utvecklas. Väl hanterade krav minskar också kostnader för framtida förbättrings- eller rättningsaktiviteter.
Konkurrensfördelar	Kravhantering bidrar till att leverera bättre produkter eller tjänster, som uppfyller affärsbehov och förväntningar. Bättre produkter eller tjänster hjälper till att erhålla förtroende och lojalitet från önskade målgrupper och ge mer konkurrensfördelar.

Fokus

© SQEB – Software Quality Engineering Board

Krav	Förstå de grundläggande koncepten kopplade till krav, deras klassifikation och abstraktionsnivåer; förklara innebörden kravattribut och kvalitetskriteriers funktion.
Kravhanteringsprocessen	Förklara kravhanteringsprocessen, dess aktiviteter, aktörer och leverabler, viktigaste principer och best practices för utveckling och hantering av krav för mjukvaru-/hårdvaruprodukter och tjänster.
Standarder, normer och juridiska bestämmelser	En översikt till de viktigaste standarderna, normerna och juridiska bestämmelserna och deras påverkan på kravhantering.
Kravhantering i utvecklingsmodeller	Förklara principerna för att anpassa den generiska kravhanteringsprocessen till specifika processmodeller.
Verktyg och tekniker	Förklara användande och fördelar med tekniker för att identifiera krav; behovs- och lösningsmodellering; kvalitetssäkring och styrningstekniker; verktygsstöd för kravhanteringsprocessen.
Övningar	Skriva bra krav; kvalitetskontroll av krav; identifiering och analys av krav.

Tabell 1 Mål med programmet på grundnivå, dess fördelar och huvudfokus

Målgruppen för certifiering på grundnivån är den som är på något sätt involverad i produkt-/affärslösningsutveckling och förvaltning, inklusive affärs- och systemanalytiker, marknadsföringsteam, hårdvaru-/mjukvaruutvecklare, IT-arkitekter, projektledare, kundrepresentanter, förvaltare och teknisk personal, IT-revisorer och kvalitetsansvariga. Att vara certifierad kravhanterare på grundnivå i programvarutestning innebär också att man kan fortsätta till högre certifieringsnivåer inom kravhantering

Huvudsyftet med programmet på grundnivå är att tillhandahålla en gemensam terminologi och en gemensam förståelse för grundkoncepten kopplat till kravhanteringsprocessen. Den kunskapsbas som tillhandahålls i programmet på grundnivå stödjer definitioner och koncept kopplat till kravhantering och den förklarar kravhanteringsprocessen tillsammans med dess leverabler. Programmet baseras på allmänt vedertagna standarder och best practices.

Inlärningsmål

Inlärningsmålen i denna kursplan har delats in i kognitiva kunskapsnivåer (K-nivåer), vilket gör det möjligt för eleven att hitta ”kompetensnivån” för varje punkt.

De kognitiva nivåerna är angivna i varje kapitel i kursplanen och klassificeras enligt följande:

- K1 – Färdighet/kunskap: Eleverna kan minnas och återge kunskap.

© SQEB – Software Quality Engineering Board

- K2 – Förståelse: Eleverna kan förklara eller sammanfatta fakta i sina egna ord, ge exempel, förstå sammanhang, tolka uppgifter.
- K3 - Tillämpa: Eleverna har förmåga att tillämpa sina kunskaper i nya specifika situationer, till exempel genom att tillämpa vissa regler, metoder eller rutiner.

Målsättning

Efter att ha genomgått programmet REQB Certified Professional for Requirements Engineering - grundnivå, kan en person:

- BO01 Förmedla de grundläggande koncepten för kravhantering, krav och dess roll för en produktlivscykel
- BO02 Skapa medvetenhet om betydelsen av kravhanteringsprocessen och dess leverabler
- BO03 Förmedla huvudaktiviteterna och syftet med kravhantering och dess delar kravledning och kravutveckling
- BO04 Förmedla grundläggande färdigheter och kompetens som krävs av en kravhanterare
- BO05 Förmedla betydelsen av kravhantering för att lyckas med ett utvecklings- och/eller förvaltningsprojekt
- BO06 Förmedla möjligt användande av olika tekniker för kravinsamling
- BO07 Förmedla vikten av spårbarhet och prioritering och deras betydelse för kravhanteringsprocessen
- BO08 Förmedla användandet av modellering för att skapa en affärslösning för ett givet affärsproblem.
- BO09 Förmedla vikten av kvalitetssäkring och kvalitetskontroll för kravhanteringsprocessen.
- BO10 Förmedla likheter och olikheter i kravhanteringsprocessen mellan vanliga modeller relaterade till utvecklings-/förvaltningsprocessen.

Detaljeringsnivå

Detaljeringsnivån i denna kursplan tillåter internationell jämförbar undervisning och examination. För att uppnå detta mål, omfattar kursplanen följande:

- Generella undervisningsmål som beskriver avsikten med grundnivån
- Inlärningsmål för varje kunskapsområde som beskriver de kognitiva kunskapsmålen för kursen och vilka kvalifikationer deltagaren ska uppnå
- En lista med information som skall läras ut och som innehåller beskrivningar och referenser till ytterligare källor, såsom vedertagen teknisk litteratur, normer och standarder, om tillämpligt
- En beskrivning av nyckelområden att lära ut, vilket också inbegriper erkända källor och standarder

© SQEB – Software Quality Engineering Board

- En lista av begrepp som studenterna måste kunna komma ihåg och förstå. De individuella begreppen beskrivs i detalj i dokumentet ”Standars Glossary of Terms used in Requirements Engineering” från REQB och motsvarande ordlista på svenska

Kursplanens innehåll är inte en beskrivning av hela kunskapsområdet kravhantering, utan speglar den kunskapsnivå som måste täckas in av kursen för grundnivån.

Hur kursplanen är organiserad

Kursplanen innehåller sju kapitel.

Översta rubriken för varje område visar vilket ämne som täcks i kapitlet och specificerar också tiden för varje kapitel.

De inlärningsmål som ska uppfyllas för varje kapitel listas i början av kapitlet.

Varje kapitel innehåller ett antal avsnitt som också har inlärningsmål och ungefärlig tidsomfattning beskriven. De underavsnitt som inte har tidsangivelser specifikt utskrivna inkluderas i tiden för överordnat avsnitt.

Till exempel:

1 Introduktion till krav	90 minuter

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska kunna påvisa efter slutförandet av respektive avsnitt.

1.1 Definition av krav

LO-1.1.1 Förklara begreppen problem, lösning och produkt (K2)

Huvudrubriken visar att minst 90 minuter ska schemaläggas för att lära ut materialet i kapitlet.

Inom varje kapitel kan det finnas flera avsnitt som alla har specifika inlärningsmål. Innehållet i ett specifikt avsnitt måste göra det möjligt att uppnå det relevanta inlärningsmålen.

© SQEB – Software Quality Engineering Board

Ändringshistorik

Version	Date	Comment
1.0	2011-11-24	Första utgåva
1.1	2012-01-18	Modifieringar
2.0	2014-06-30	Andra utgåva

© SQEB – Software Quality Engineering Board

1 Introduktion till krav	110 minuter

Begrepp

Begränsning, Funktionellt krav, Icke-funktionellt krav, Krav, Kravhantering, Kravledning, Kravutveckling, Kritikalitet, Kvalitetsattribut till krav, Lösning, Prioritet, Produkt, Produktkrav, System, Teknisk begränsning, Validering, Verifiering, Verksamhetsbegränsning, Verksamhetsproblem, Åtagande

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska kunna påvisa efter slutförandet av respektive avsnitt.

1.1 Definition av krav

1.1.1 Definition av ett problem, en lösning och en produkt

LO-1.1.1 Förklara begreppen problem, lösning och produkt (K2)

1.1.2 Definition och klassificering av krav

LO-1.1.2 Förklara innebörden av begreppet krav och klassificeringar av krav (t.ex. produktkrav och begränsningar) (K2)

LO-1.1.3 Förklara skillnaden mellan de olika nivåerna av krav (dvs. verksamhetskrav, lösnings-(system-)krav, och produkt-(komponent-)krav) och deras innebörd i kravhanteringen (K2)

1.1.3 Vanliga kravattribut

LO-1.1.4 Förklara innebörden hos vanliga kravattribut (t.ex. åtagande, prioritet, kritikalitet) (K2)

1.1.4 Kravkvalitet

LO-1.1.5 Förstå vanliga problem rörande krav och förklara den roll som validering, verifiering och kvalitetskriterier har för att förbättra kvaliteten hos krav. (K2)

1.2 Standarder och normer

© SQEB – Software Quality Engineering Board

LO-1.2.1 Förklara möjliga fördelar och tillämpning av standarder för kravhantering. (K2)

© SQEB – Software Quality Engineering Board

1.1 Definition av krav	90 minuter

För att förstå disciplinen kravhantering och dess huvudsakliga in- och utdata är det viktigt att visa sammanhanget och introducera de viktigaste termerna och definitionerna.

Hela listan med termer och definitioner kan återfinnas i REQB:s ”SQEB Engelsk/svensk ordlista, Kravhantering”.

Syftet med detta kapitel är att tillhandahålla grundläggande definitioner för kravhantering och hur dessa relaterar till kravhantering i sig.

1.1.1 Definition av problem, lösning och produkt

Huvudsakligt indata till kravhanteringsprocessen är verksamhetsproblem, verksamhetsbehov och/eller affärsmål. Ett verksamhetsproblem är en beskrivning av vad en kund önskar göra för att förverkliga eller förbättra sina verksamhetsprocesser. Problemet hjälper till att beskriva en kunds verksamhetsbehov.

Ett av huvudmålen med kravhantering är att definiera verksamhetslösningar för givna verksamhetsproblem, -behov eller -mål. En lösning är svaret på kundens behov. Dessa behov uttrycks vanligen som kundkrav och är inflöde till kravhantering. En lösning kan vara ett system eller en systemkomponent, ny eller ändrad funktionalitet, ändrad konfiguration, processförbättring etc. Generellt är en lösning en detaljering av krav på högre nivå.

Resultatet av realiseringsprocessen kallas produkt.

För kurshållare: förklara innebörden av problem, lösning och produkt med exempel från verkliga livet från ett utvecklings- eller förvaltningsprojekt.

1.1.2 Definition och klassificering av krav

Enligt IEEE 610 kan ett krav definieras som:

1) Ett villkor eller en egenskap som en intressent behöver för att lösa ett problem eller uppnå ett mål.
2) Ett villkor eller en egenskap som ett system eller en systemkomponent måste inneha för att uppfylla ett kontrakt, en standard, en specifikation eller ett annat formellt styrande dokument.
3) En dokumenterad förekomst av ett villkor eller en egenskap beskrivet i (1) eller (2).

Ett huvudsyfte med krav är att uttrycka kundens behov och förväntningar rörande den planerade lösningen. Krav fungerar också som en grund för bedömning, planering, genomförande och

© SQEB – Software Quality Engineering Board

övervakning av projektaktiviteter och är ofta en del av serviceavtal, beställningar eller kontrakt. Krav definierar inte bara vad som ska göras utan de etablerar lösningens gränser, leveransens omfattning, leveransplaner och kontraktstjänster.

I allmänhet kan krav klassificeras som produktkrav eller begränsningar.

Produktkrav beskriver egenskaper, funktioner, värden och tjänster som krävs av en given produkt och består av funktionella och icke-funktionella krav. Funktionella krav beskriver funktionaliteten (beteendet) hos produkten, medan icke-funktionella krav beskriver kvalitetsattribut hos produkten (även kallat kvalitetsmål).

Skillnaden mellan funktionella och icke-funktionella krav kan uttryckas genom följande förklaringar:

- Funktionella krav beskriver *vad* lösningen gör.
- Icke-funktionella krav beskriver *hur* lösningen utför sina funktioner.

Begränsningar är restriktioner rörande utvecklingsprocessen, ett systems funktioner eller dess livscykel. Det finns två typer av begränsningar: verksamhetsbegränsningar och designrestriktioner.

- Verksamhetsbegränsningar beskriver restriktioner rörande projektets flexibilitet att implementera den efterfrågade lösningen (till exempel finansiella restriktioner eller tidsrestriktioner, begränsningar rörande tillgängliga resurser, kompetens i projektteamet, andra organisatoriska begränsningar, branschspecifik lagstiftning, specifika standarder och regelverk)
- Designrestriktioner är alla begränsningar som rör lösningens arkitektur (till exempel plattform för hård- och mjukvara, programmeringsspråk eller -teknik, programvara som måste användas, databasstorlek, resursutnyttjande, meddelandestorlek och timing, storlek på programvara, maximalt antal och storlek på filer, poster och dataelement).

För organisationer som arbetar med en produkt kan ett krav vara extern eller internt – vissa krav kan komma från kunderna och andra från organisationen som utvecklar produkten. Krav som kommer från olika källor ska analyseras tillsammans.

Krav kan delas in i olika typer, vilka visar att de behandlar olika aspekter av produkten eller dess utvecklingsprocess. Krav kan klassificeras i abstraktionsnivåer vilka visar olika detaljeringsnivåer; från verksamhetskrav på hög nivå till detaljerade krav rörande programvaran eller systemets funktioner, eller till och med delar av lösningsdesignen (t.ex. prototyper av skärmbilder eller tangentbord för mobiltelefoner).

Nedanstående visar några vanligt förekommande abstraktionsnivåer för krav:

- Verksamhetskrav

© SQEB – Software Quality Engineering Board

  - Högnivåkrav som beskriver vad verksamheten önskar uppnå men inte hur det ska implementeras. Dessa krav beskriver kundernas önskemål, behov och förväntningar och kallas ofta verksamhets- eller kundkrav på hög nivå.
- Lösnings-/systemkrav
  - Detaljering av kundens krav som beskriver lösningen som är en av flera möjliga sätt att uppfylla kundens krav. Lösningen kan innehålla krav som är icke IT-relaterade för processförändringar eller förändringar av organisation/roller. Lösning-/systemkrav är kundens krav uttryck i mer tekniska termer som kan användas för designbeslut.
- Produkt-/komponentkrav
  - Funktioner och egenskaper hos lösningen. En komplett specifikation av en produktkomponent, inklusive format, funktion, prestanda, etc. Denna abstraktionsnivå saknas ofta på grund av att det anses vara en del av lösningsdesignen.

När man arbetar med krav på olika nivåer är det viktigt att etablera och underhålla spårbarhet (se avsnitt 4.3 Spårbarhet hos krav). Spårbarhet är en koppling mellan projektartefakter på olika nivåer, till exempel mellan verksamhetskrav och lösningskrav eller mellan krav och relaterade testfall.

För kurshållare: tillhandahåll exempel på krav av olika typer och på olika abstraktionsnivåer.

1.1.3 Vanliga kravattribut

De viktigaste attributen för ett högnivåkrav är följande:

- Åtagande
- Prioritet
- Kritikalitet

Åtagande är graden av skyldighet att uppfylla kravet. Det definieras vanligen genom nyckelord som tilldelas högnivåkraven (”ska”, ”bör”, ”kan”) [efter ISO/IEC/IEEE 29148:2011, tidigare IEEE 830]. I vissa fall används begreppet ”ska inte” som nyckelord för att uttrycka aspekter som lösningen inte får göra. Nyckelorden”bör”, ”kan” relaterar vanligen till verksamhetskrav innan överenskommelse. Efter överenskommelse och etablering av baseline ska kravens nyckelord kunna avgöra graden av skyldighet in mer strikta termer: ”kommer att” eller ”ska”.

Graden av skyldighet att uppfylla ett krav kan också uttryckas genom att använda MoSCoW-notation (Måste ha (Must), Bör ha (Should), Kan ha (Could), Ska inte ha denna gång men kommer att ha i framtiden (Won´t this time)).

Så snart kunden och leverantören av lösningen nått en överenskommelse kan projektdeltagarna göra ett gemensamt åtagande att uppfylla kraven.

© SQEB – Software Quality Engineering Board

Krav kan utvecklas under projektets gång. Vartefter kraven utvecklas kommer ett gemensamt åtagande från projektdeltagarna rörande de ändrade kraven att säkerställa åtagande för de resulterande ändringar som uppstår i projektplaner, aktiviteter och arbetsprodukter.

En av de viktigaste innebörderna i ett åtagande av ett krav är ansvar för de slutliga produkterna. Det kan finnas juridiska skyldigheter relaterade till produktens kvalitet. Juridiskt ansvar är ofta relaterat till ett specifikt krav (till exempel ett miljökrav för ett kärnkraftverk eller ett säkerhetskrav för ett flygplan) som måste uppfyllas för produkten som specificeras. Ansvaret kan också röra fel i produkten. Åtagande är ett sätt att uttrycka uppfyllande av legala krav.

Juridiskt ansvar ska definieras i kontraktet mellan leverantören och kunden. Vissa branscher kan också ha krav på sig att uppfylla kontraktskrav eller juridiska krav eller branschspecifika standarder.

Kravattributet prioritet uttrycker viktigheten/angelägenheten hos ett krav. Enligt [SWEBOK] gäller i allmänhet att ju högre prioritet ett krav har, desto mer väsentligt är kravet för att uppfylla produktens övergripande mål. Ofta klassificeras prioriteten enligt en fast skala som till exempel obligatoriskt, mycket önskvärt, önskvärt, eller valfritt och måste ofta ställas mot kostnaden för utveckling och implementering.

Ett kravs kritikalitet är resultatet av en bedömning av risken för den skada som skulle uppstå om kravet inte uppfylldes. Kritikaliteten uttrycks i nivåer; ju högre nivå, desto högre allvarligare konsekvenser vid funktionella fel.

I vissa fall kan kravattribut också uttryckas med hjälp av Kano-modellen. Denna modell fokuserar på att särskilja produktfunktionalitet enligt följande:

- Basattribut – detta är egenskaper som produkten måste ha för att uppfylla kundens behov.
- Förväntade attribut – en förmåga, kunskap, färdighet eller beteendemässig egenskap som kan sammankopplas med det sätt som produkten kommer att leverera sin förmåga till användaren. Prestandaattribut förbättrar kundnöjdheten.
- Sensationella attribut – attribut som är oförutsedda av kunden (ibland kallade ”omedvetna behov”) som är önskvärda att upptäckas av leverantören. De kan utföra en märkbar konkurrensfördel.

Kano-modellen kan användas för att uttrycka både prioritet och åtagande.

Utöver ovanstående attribut kan krav kategoriseras enligt följande [ISO/IEC/IEEE 29148:2011, tidigare IEEE 1233]:

- Identifiering
- Genomförbarhet
- Risk
- Källa
- Typ

© SQEB – Software Quality Engineering Board

För kurshållare: förklara de vanliga attributen hos krav och ge några exempel för att visa deras innebörd.

1.1.4 Kravkvalitet

Kravhantering är en av de viktigaste framgångsfaktorerna i ett projekt. Krav är grunden för fortsatt arbete, därför är det väldigt viktigt att säkerställa bästa tänkbara kvalitet hos krav och andra produkter i kravhanteringsprocessen. De viktigaste frågorna man måste hantera när man arbetar med krav är följande:

- Otydliga verksamhetsmål för kraven eller själva projektet
- Kommunikationsproblem (ofta ett resultat av språk- och/eller kunskapsbarriärer, inklusive brist på kunskap inom verksamhetsområdet), vaga formuleringar (ofta ett resultat av otillräcklig eller omätbar definition av ett krav, eller brist på gemensam ordlista)
- Instabila krav (ofta orsakat av oklara verksamhetsmål för kraven eller projektet i sig)
- Dålig kvalitet hos kraven (se kvalitetskriterier för krav nedan)
- Förgyllning (tillägg av extra funktioner som egentligen inte behövdes)
- Otillräckligt intressentengagemang
- Saknade intressenter (ofta ett resultat av en bristfällig kravhanteringsprocess)
- Dålig planering (ofta orsakad av otydliga eller saknade verksamhetsmål för kraven eller för projektet i sig)
- Otillräckliga krav- eller lösningsspecifikationer (vilket resulterar i luckor och saknade eller fragmenterade krav)

Några av problemen som listas ovan kan lösas rätt enkelt genom att applicera kvalitetskriterier för kraven. Enligt Wiegers definition av vanliga kvalitetskriterier för krav måste varje krav vara:

- Korrekt – kravet måste på ett exakt sätt beskriva de funktioner som ska tillhandahållas. Referenspunkten för att bedöma korrektheten är källan till kravet (till exempel kund eller ett systemkrav på högre nivå).
- Genomförbart – kravet måste vara möjligt att implementera inom ramen för känd kapacitet och/eller begränsningar för systemet och miljön.
- Nödvändigt – kravet ska beskriva vad kunden (eller andra intressenter) verkligen behöver och vad som behövs för att uppfylla ett externt krav eller gränssnitt eller en specifik standard.
- Prioriterat – kravet ska tilldelas en prioritet som indikerar hur viktigt det är för en specifik produktrelease.

© SQEB – Software Quality Engineering Board

- Entydigt – kravet ska bara kunna tolkas på ett sätt. Kravets olika läsare ska kunna tolka och förstå kravet på samma sätt.
- Verifierbart – det ska gå att verifiera att kravet har implementerats korrekt.
- Unikt – kravet ska inte innehålla flera krav; detta kräver tillräcklig detaljering för att specificera ett enskilt krav.
- Design- (implementations-)oberoende – kravet ska beskriva “vad som ska göras”, inte “hur det ska göras”. Innehållet i ett krav ska inte föreskriva eller antyda implementeringsdetaljer.

Kvalitetskriterier kan inte appliceras enbart på ett enskilt krav, utan kan även användas för en kravspecifikation.

De huvudsakliga kvalitetskriterierna tillstår att kravspecifikationen måste vara:

- Fullständig – inget krav eller nödvändig information ska saknas i kravspecifikationen. Fullständighet är också en önskvärd egenskap hos varje enskilt krav och detaljeringsnivå.
- Konsistent – de krav som beskrivs i en specifikation får inte stå i konflikt med andra produktkrav eller med system- eller verksamhetskrav på en högre nivå.
- Modifierbar – det måste vara möjligt att ändra kraven i specifikationen. En historik över genomföra ändringar för varje krav måste upprätthållas.
- Spårbar – varje krav ska vara möjligt att länka till sin källa (till exempel, ett systemkrav på högre nivå, ett användningsfall, eller en kundbeskrivning) och till relaterade implementationsartefakter (till exempel designelement, källkod och testfall).

För kurshållare: förklara generella kvalitetskriterier för krav och tillhandahåll några exempel av krav som uppfyller respektive inte uppfyller kvalitetskriterierna.

Att säkerställa den kvalitetsnivå som krävs för krav eller andra produkter från kravhanteringsprocessen kan stödjas genom validering och verifiering.

Enligt CMMI visar valideringsaktiviteter att om en produkt eller produktkomponent uppfyller sin avsedda användning i sin avsedda miljö så du vet att du ”bygger rätt sak”. Kunder tar ofta fram otydliga produktbeskrivningar eller krav; validering underlättar då förståelsen för vad behovet egentligen är (genom användandet av verktyg såsom scenarier, användningsfall, prototyper etc.)

Validering utförs vanligen med hjälp av en kund eller intressent hos kunden och syftar till att bekräfta att kraven eller kravspecifikationen faktiskt beskriver vad kunden behöver.

Enligt CMMI tillhandahåller verifiering kontrollpunkter för vilka utvalda produkter eller delprodukter som har verifierats för att bekräfta att de uppfyller kraven. Verifieringsaktiviteter fokuserar på inkrementell bekräftelse av implementeringen av kraven; de möjliggör tidig och kontinuerlig bekräftelse av att produkten byggs rätt.

Verifiering är en jämförelse av en delprodukt med dess specifikationer. Det kan därmed fastställas avgöras om produkten har utvecklats rätt och om specifikationerna som fastställdes under

© SQEB – Software Quality Engineering Board

föregående fas har uppfyllts. De vanligaste teknikerna för validering och verifiering är granskning, statisk analys och dynamisk testning.

Skillnaden mellan validering och verifiering kan uttryckas på följande sätt:

- Verifiering – skapade vi produkten rätt?
- Validering – skapade vi rätt produkt?

1.1.5 Kravhantering, kravledning och kravutveckling

Kravhantering (RE) kan definieras som en underdisciplin till systemutveckling, vilken fokuserar på att fastställa, utveckla, underhålla och hantera ändringar till krav på hårdvaru- och mjukvarusystem [SWEBOK, Sommerville]. Enligt CMMI består kravhantering av kravledning och kravutveckling.

Syftet med **kravledning** är att hantera baselines av överenskomna uppsättningar krav på lösningen och att säkerställa överensstämmelse mellan dessa krav och projektets planering och arbetsprodukter.

Kravledning består både av ett arbetsramverk för kravhantering och av stödjande processer för kravutvecklingsprocessen. Kravledningen andra roll är att etablera och tillhandahålla gränssnitt gentemot andra utvecklings- och ledningsprocesser som gränsar mot kravhantering såsom projektledning, riskhantering, design, konfigurationshantering, ändringshantering och kvalitetssäkring.

Kravutveckling är en uppsättning aktiviteter, uppgifter, tekniker och verktyg för att identifiera, analysera, dokumentera och validera krav på olika abstraktionsnivåer. Det omfattar processen att omvandla behov till krav likväl som utveckling (nedbrytning av högnivåkrav) av en lösning för dessa krav.

En bra kravhanteringsprocess är en av de största framgångsfaktorerna i projekt. Väl insamlade, analyserade, dokumenterade och hanterade krav minskar projektrisker då de utgör en tydlig och begriplig grund för lösningsdesign. Förståelse för vikten av bra krav och en bra kravhanteringsprocess finns dock fortfarande inte alltid, eller så förbises det helt enkelt.

© SQEB – Software Quality Engineering Board

Följande kan vara skäl för att kravhantering försummas:

- Stor tidspress och högt kostnadstryck
- Inriktning enbart på snabba resultat
- Hänsyn tas enbart till de funktionella kraven
- Brist på förståelse för kravhanteringens vikt för ett lyckat projekt

En bristfällig kravhanteringsprocess kan få följande konsekvenser:

- Krav av låg kvalitet
- Krav som förändras ofta under produktutvecklingen
- Krav som inte uppfyller acceptanskriterierna och/eller inte levererar ett värde till kunden eller andra intressenter
- Saknade krav eller designrestriktioner

© SQEB – Software Quality Engineering Board

1.2 Standarder och normer	20 minuter

Notera: Standardernas identifikationsnummer behöver inte memoreras.

Det finns många standarder och processnormer som är användbara inom kravhantering. Några av dem tillhandahåller processmodeller för lösningsutveckling, andra tillhandahåller riktlinjer för framtagande av olika kravspecifikationer eller för klassificering av krav.

Ett huvudsyfte med standarder är att få en nationell och internationell överensstämmelse av produkter och processer. Standarder standardiserar utvecklingsmetoder och -leverabler, tillhandahåller gemensam terminologi och underlättar kommunikation rörande både verksamhet och teknik.

De viktigaste standarderna och normerna som används i REQB:s kursplan listas nedan.

Standarder:

- ISO 9000. Standardfamiljen ISO 9000 tillhandahåller krav för ett kvalitetsstyrningssystem och definierar koncept och grunder hos ett kvalitetsstyrningssystem (QMS).
- ISO/IEC 25000 (tidigare ISO 9126). Här definieras en kvalitetsmodell med sex huvudkategorier (funktionalitet, tillförlitlighet, användbarhet, effektivitet, underhållbarhet, portabilitet) vilka kan vara en grund för insamling, specificering eller validering och verifiering.
- IEEE Standard 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology
- ISO/IEC/IEEE 29148–2011. Systems and software engineering -- Life cycle processes -- Requirements engineering. Har ersatt följande standarder:
    - IEEE Standard 830-1998. IEEE Recommended Practice for Software Requirements Specifications (also known as Recommended Practice for SRS).
    - IEEE Standard 1233-1998. IEEE Guide for Developing System Requirements Specifications (också känd som Guide for Developing of SyRS).
    - IEEE Standard 1362-1998. IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document
- SWEBOK - The Guide to the Software Engineering Body of Knowledge (känd som ISO Technical Report 19759). SWEBOK beskriver allmänt vedertagen kunskap om programvaruutveckling. Dess 10 kunskapsområden sammanfattar grundläggande koncept och innehåller en referenslista som hänvisar till mer detaljerad information. Ett av områdena tillägnas kravhantering
- SEBOK – The Guide to the System Engineering Body of Knowledge.

Processnormer för utveckling:

© SQEB – Software Quality Engineering Board

- ISO 12207. Standard for Software Life Cycle Process.
- ISO 15288. System Life Cycle Process.

Båda kan användas för att organisera utvecklingsprocessen för lösningen.

Processnormer för utvärdering och förbättring av processer:

- ISO 15504 Information Technology — Process assessment, även känd som SPICE (Software Process Improvement and Capability Determination).
- CMMI. Capability Maturity Model Integration.

Båda stödjer processförbättring och kan användas för processbedömning och förbättring. De definierar viktiga områden för kravhantering.

For kurshållare: förklara varför dessa standarder är relevanta och viktiga när man definierar en kravhanteringsprocess.

© SQEB – Software Quality Engineering Board

2 Kravhanteringens sammanhang	40 minuter

Begrepp

Verksamhetsanalys

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska kunna påvisa efter att slutförandet av respektive avsnitt.

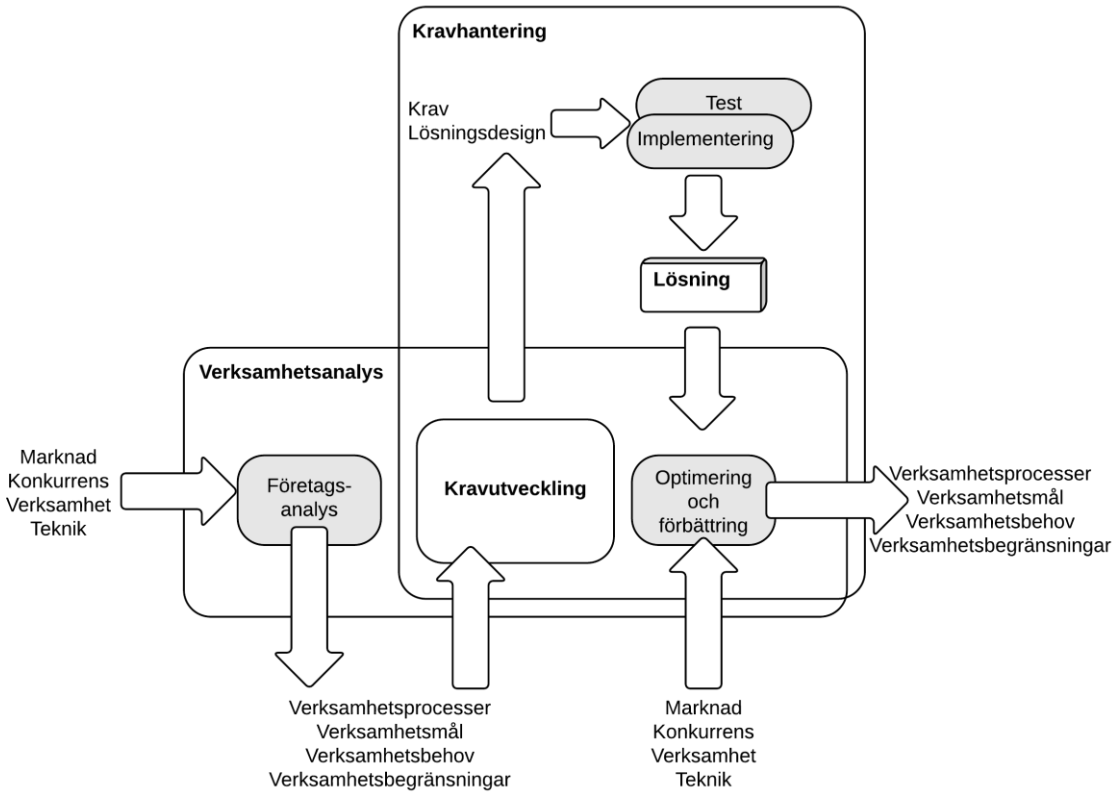
2.1 Kravhantering i sitt sammanhang

LO-2.1.1 Förklara kravhanteringens sammanhang och hur kravhantering är en del av lösningsutveckling och livscykeln för förvaltning (K2)

© SQEB – Software Quality Engineering Board

2.1 Kravhantering i sitt sammanhang	20 minuter

Kravhantering utförs inte isolerat från andra aktiviteter. Den är sammanlänkad med andra discipliner (figur 1) och ingår i utvecklingsprocessen.



Figur 1 Kravhanteringens sammanhang

Utgångspunkten för kravhantering är verksamhetsanalys. För att kunna föreslå den bästa lösningen för ett givet verksamhetsproblem är det viktigt att definiera problemet korrekt [IBAQB].

Verksamhetsanalys är en disciplin som tillhandahåller en uppsättning aktiviteter, verktyg och metoder med syfte att fastställa verksamhetens behov, problem och mål samt bestämma lämplig lösning för att tillfredsställa dessa behov och lösa angivna verksamhetsproblem.

Dessa verksamhetslösningar kan innefatta utveckling av programvarusystem eller programvarukomponenter, vidareutveckling av existerande programvara, förbättringar av verksamhetsprocessen, organisationsförändringar, etc. Verksamhetsanalysen definierar verksamhetsproblemen som ska lösas av en specifik lösning vilket blir indata för fortsatt kravhantering. Verksamhetsmål och -behov fastställs genom verksamhetsanalys och utvecklas till krav på en lösning. Därför kan kravhantering ses som en fortsättning eller en del av

© SQEB – Software Quality Engineering Board

verksamhetsanalysprocessen. Utdata från kravhantering är lösningsdesignen, som sedan implementeras, testas och slutligen levereras till kunden.

Den implementerade lösningen förblir sällan oförändrad under en längre tidsperiod. Marknaden och verksamheten förändras, teknologin utvecklas hela tiden och nya kopplingar inom verksamheten uppstår. Det är därför verksamhetsanalys och kravhantering fortsätter genom hela lösningens livscykel. Efter leverans av lösningen används verksamhetsanalysen för att sträva efter förbättringar som kan utöka marknaden, förbättra effektiviteten eller vinna konkurrensfördelar. Nya verksamhetsbehov kommer att uppstå. Dessa verksamhetsbehov utvecklas genom kravhantering för att föreslå nya, förbättrade lösningar eller utöka befintlig lösning med nya, bättre funktioner.

För kurshållare: Förklara kravhanteringens sammanhang med verkliga exempel. Tillhandahåll exempel på relationer mellan kravhantering och verksamhetsanalys (i form av olika produkter och aktiviteter).

© SQEB – Software Quality Engineering Board

2.2 Relaterade processer	20 minuter

Kravhantering, och särskilt den del som kallas kravledning, verkar i ett större sammanhang och har starka kopplingar till följande processer:

- Verksamhetsanalys – Som förklarat i tidigare avsnitt är verksamhetsanalys och kravhantering starkt sammankopplade.
- Projektledning – Projektledare ska inkludera kravspecifika aktiviteter i sin projektledningsplan, avsätta erforderlig tid och resurser och säkerställa att de utförs på rätt sätt. Kravhanterare kan behöva skapa en kravplan vilken specificerar de aktiviteter som behövs för att färdigställa kravspecificeringen.
- Riskhantering – Vissa krav kan medföra projekt- eller produktrisker vilka ska hanteras via riskhanteringsprocessen.
- Analys och design – Krav är obligatoriska indata vid analys- och designaktiviteter. Spårbarhet krävs mellan krav och analys- och designprodukter.
- Konfiguration och ändringshantering – Krav ska konfigurationshanteras och följa processen för konfigurationshantering. Ett krav kan kopplas till ett konfigurationshanterat objekt.
- Testning – Testning ger återkoppling rörande skillnader mellan angivna krav och produktens utvecklade funktionella och/eller icke-funktionella attribut. Spårbarhet mellan krav, test och andra artefakter är nödvändigt för att tydligt synliggöra testtäckning. Beroende på associerade testobjekt kan ett specifikt krav ha en status som visar om det är implementerat eller inte.
- Leveranshantering – Kravhanteringens resultat är utgångspunkten för leveranshantering. För varje lösning eller produkt som levereras måste det vara möjligt att identifiera vilka krav som implementeras i den aktuella leveransen.

För olika utvecklingsmodeller, eller olika produkter, kan kravhantering kopplas även till andra processer.

För kurshållare: tillhandahåll exempel på kopplingar mellan kravhantering och andra discipliner (I form av olika produkter och aktiviteter).

© SQEB – Software Quality Engineering Board

3 Introduktion till kravhanteringsprocessen	80 minuter

Begrepp

Generisk kravhanteringsprocess, Generisk kravutvecklingsprocess, Intressent, Kravhanterare, Kravledare, Kravutvecklare, Kund, Leverantör

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska kunna påvisa efter att slutförandet av respektive avsnitt.

3.2 Generisk kravhanteringsprocess

LO-3.2.1 Förklara de vanligaste målen med kravhantering och vanliga aktiviteter i kravhanteringsprocessen (K2)

3.3 Roller och ansvarsområden

LO-3.3.1 Förklara rollerna inom kravledning och kravutveckling (K2)

LO-3.3.2 Komma ihåg de grundläggande rollerna inom kravhantering och kunna förklara deras påverkan på kravhanteringsprocessen (K1)

LO-3.3.3 Komma ihåg de viktigaste färdigheter och kompetenser som krävs av en person som är involverad i kravhantering (K1)

© SQEB – Software Quality Engineering Board

3.1 Introduktion till kravhanteringsprocessen	40 minuter

Strukturen hos kravhanteringsprocessen beror på olika faktorer, såsom organisationens kultur och mognad, vilken utvecklingsmodell som används, etc. REQB använder följande kategorisering:

- Generisk kravhanteringsprocess
- Kravhanteringsprocess i utvecklings- och förvaltningsmodeller och angreppssätt
- Kravhanteringsprocess i mognadsmodeller

Den generiska kravhanteringsprocessen ska utgöra startpunkten för alla organisationer som är involverade i utvecklingsarbetet av lösningen eftersom den tillhandahåller de mest avgörande processerna för att hantera krav. Denna generiska kravhanteringsprocess kan anpassas till en organisations specifika behov och den utvecklings- eller förvaltningsmodell som ska tillämpas.

Resultatet av denna anpassning är en kravhanteringsprocess för modeller och metoder inom utveckling och förvaltning. Utöver processmodeller för utveckling finns även mognadsmodeller. Dessa mognadsmodeller används främst för att utvärdera den nuvarande mognaden i specifika organisationer och för att införa förbättringar som är nödvändiga för att öka organisationens effektivitet. Vi kan därför definiera en annan variant av den generiska kravhanteringsprocessen – kravhanteringsprocess i mognadsmodeller. Denna variant av kravhanteringsprocess täcks av REQB Advanced Level program.

Den generiska kravhanteringsprocessen bör anpassas mot behoven i en organisation med hänsyn taget till vilken sorts lösning som utvecklas, verksamhetsbransch, verksamhetsprocesser, organisationskultur och nivå på kompetens och färdigheter hos personalen som ansvarar för kravhanteringsprocessen.

För kurshållare: förklara REQB:s synsätt på kravhanteringsprocessen och visa behovet av att anpassa den allmänna kravhanteringsprocessen för särskilda behov.

© SQEB – Software Quality Engineering Board

3.2 Generisk kravhanteringsprocess	20 minuter

Kravhantering är en disciplin som inkluderar processer för insamling, strukturering och hantering av krav. Specifika aktiviteter som täcks in av den allmänna kravhanteringsprocessen är följande:

- Kravinsamling
- Kravanalys
- Kravspecificering
- Validering och verifiering av krav
- Kravspårbarhet
- Konfigurations- och ändringshantering
- Kvalitetssäkring

Kravhanteringsprocessen är en strukturerad uppsättning av ovannämnda aktiviteter. Aktiviteterna sorteras in under en kravledningsprocess eller en kravutvecklingsprocess, beroende på syftet och skedet av lösningsutvecklingen. En komplett processbeskrivning bör inkludera kopplingar till andra relaterade discipliner och områden (till exempel verksamhetsanalys och testning), indata till och resultat från aktiviteterna, information rörande ansvar för specifika aktiviteter och deras resultat, erforderlig kompetens och de verktyg som behövs för att stödja aktiviteterna (mer detaljer i kapitel 7 – Verktygsstöd).

Huvudaktiviteter, indata och resultat för de specifika kravhanteringsprocesserna presenteras i tabell 2.

© SQEB – Software Quality Engineering Board

Process	Underlag	Huvudaktiviteter	Resultat
Kravutveckling			
Kravinsamling	Verksamhetsbehov Inledande beskrivning av omfattningen Intressenter	Samla in krav från intressenter Detaljera kända högnivåkrav Exkludera icke nödvändiga funktioner	Verksamhetskrav Begränsningar Antaganden
Kravanalys	Verksamhetskrav Designrestriktioner Antaganden Intressenter Organisations-tillgångar Modelleringstekniker Verktyg	Vidareutveckla verksamhetskrav till system-/lösningskrav och ner till produkt-komponentkrav Lösa motsättningar mellan krav Etablera ramar för lösningen Utveckla modeller för verksamhetslösningen	Överenskomna krav Fastställd omfattning Verksamhets-lösningsmodell
Kravspecifikation	Krav Designrestriktioner Antaganden Intressenter Organisations-tillgångar	Dokumentera krav	Kravspecifikations-dokument
Kravvalidering och -verifiering	Krav Kravspecifikations-dokument	Kontrollera kvaliteten hos krav och krav-specifikationsdokument	Validerade krav Validerade kravspecifikations-dokument Defekter Problem Ändringar
Kravledning			

© SQEB – Software Quality Engineering Board

Process	Underlag	Huvudaktiviteter	Resultat
Kravspårning	Krav Intressenter Organisations-tillgångar Verktyg	Etablera en struktur för spårbarhet Definiera och underhålla spårbarhet hos krav	Spårbarhetsmatris
Konfigurations- och ändrings-hantering	Krav Verktyg Ändringsbegäran Projektplaner Spårbarhetsmatris	Identifiera och hantera konfigurationsobjekt Efterfråga, bestämma genomförbarhet, planering, implementering och utvärdering av förändringar av ett system, dokument eller annan produkt i projektet Genomföra riskanalys Processa ändringar och säkerställa ändringars spårbarhet	Fastställda krav Resultat från riskanalys Ändrade krav och kravspecifikations-dokument Projektplaner
Kvalitetssäkring	Krav Kravspecifikations-dokument Lösningsmodeller Intressenter Organisations-tillgångar Standarder Verktyg Spårbarhetsmatris Riskanalys	Säkerställa att de olika kravhanterings-processerna och deras produkter är av god kvalitet Tillämpa relevanta standarder Utföra granskningar och andra kvalitetssäkrings-aktiviteter	Krav Kravspecifikations-dokument Verksamhetslös-ningsmodeller Defekter Problem Ändringar Förbättringar Definierade och kommunicerade kravhanteringspro cesser och -aktiviteter

Tabell 2 Översikt till kravhanteringsaktiviteter, underlag och resultat

Effektiviteten hos kravhanteringsprocessen är inte densamma i alla organisationer eller projekt. Vissa faktorer, externa eller interna, kan ha negativa influenser på den eller dess produkter.

© SQEB – Software Quality Engineering Board

Faktorer på den interna sidan (t.ex. inifrån produktleverantörens organisation) inkluderar:

- Brist på kunskap om verksamhetens eller användarens bransch
- Brist på kunskap om teknik som efterfrågas av kunden
- Ineffektiva angreppssätt/metoder/verktyg rörande kravhantering
- Otillräcklig erfarenhet, kunskap och kompetens hos personal

Faktorer på den externa sidan (t.ex. utanför produktleverantörens organisation) inkluderar:

- Brist på kommunikation eller ineffektiva angreppssätt för kommunikation rörande krav, behov eller förväntningar
- Otydliga och/eller ändrade verksamhetsmål som resulterar i instabila krav
- Otillräcklig kunskap om produktutvecklingsprocessen
- Otillräckligt engagemang av användare och/eller verksamhetsintressenter

Kravhantering behandlar många aktiviteter, beroende på sin roll i lösningsutvecklingen eller livscykeln för förvaltning, såväl som begränsningar och egenskaper hos den planerade lösningen.

Kravhanteringsprocessen och dess produkter (såsom kravspecifikationer eller verksamhetslösningsförslag) ska tillhandahålla den största nyttan för kunden och andra intressenter. För att kunna säkerställa att den planerade lösningen bäst uppfyller kundens behov och möter förväntningar ska krav utvecklas från ett kundperspektiv. Exempel på metoder från den kundorienterade kravhanteringsprocessen är följande:

- Kundorienterad analys och design
- Prototypbaserat angreppssätt
- Använda demonstrationer som ett medel för att validera systemets inkrement

För kurshållare: tillhandahåll en exempelöversikt över kravhanteringsaktiviteter, underlag och resultat för ett specifikt verksamhetsområde/-projekt.

© SQEB – Software Quality Engineering Board

3.3 Roller och ansvarsområden	20 minuter

En lyckad kravhanteringsprocess kräver tydliga definitioner av roller och ansvarsområden. Roller kan i allmänhet klassificeras som roller som påverkar eller påverkas av kravhantering samt roller inom kravhantering.

De viktigaste rollerna som påverkar eller påverkas av kravhantering är kunden och leverantören. Kunden är en person, grupp eller organisation som efterfrågar lösningen, medan leverantören är en person, grupp eller organisation som tillhandahåller lösningen. Kunden formulerar sina behov och tillhandahåller de inledande verksamhetsbehoven och – förväntningarna. Dessa tillhandahålls ofta tillsammans med anbud/tjänstebegäran. Leverantören ansvarar för att analysera dessa behov och utvinna krav från den grunden. Huvudmålet för leverantören är att leverera en lösning som uppfyller kundens behov.

Rollerna inom kravhantering kan definieras som kravledare och kravutvecklare. Båda roller kan också kallas kravhanterare.

Den stora variationen av uppgifter kopplat till kravhantering har resulterat i många olika roller för personer som utför kravhanteringsrelaterade uppgifter. Olika organisationer använder egna namn, som t.ex. kravhanterare, kravledare, kravutvecklare, verksamhetsanalytiker, systemanalytiker, lösningsarkitekt, systemarkitekt, designer etc. Dessutom finns regionala varianter av dessa roller beroende på kultur, vanor och traditioner. Problemet är att specifika ansvarsområden och kompetenser som behövs för de nämna rollerna sällan är definierade, tydliga eller förstådda.

För att tillhandahålla en gemensam förståelse för rollerna och dess ansvarsområden definierar REQB – i översättning av SQEB - rollerna kravledare och kravutvecklare. Denna indelning är ett resultat av skillnaderna i de uppgifter som utförs i kravlednings- och kravutvecklingsaktiviteterna. En kravledare är en person med det övergripande ansvaret för dokumentation, analys, spårbarhet, prioritering och samordning av överenskommelse av kraven och för att sedan övervaka ändringar och kommunicera krav till relevanta intressenter. En kravutvecklare är en person som huvudsakligen deltar i insamling, analys, dokumentation och prioritering av krav.

En kravledare och en kravutvecklare bör inneha följande färdigheter:

- Kompetens inom relevant metodik (t.ex. praktisk kunskap inom kravhanteringsprocessen, metoder, tekniker och verktyg)
- Kunskap om källor till vedertagen kunskap och de mest relevanta standarder kopplat till kravhantering.
- Precist, analytiskt och klart tänkande
- Förmåga att agera på ett strukturerat sätt
- Vara saklig och kunna förhandla
- Agera med självförtroende

© SQEB – Software Quality Engineering Board

- Förmåga att argumentera och övertyga
- Språkliga färdigheter och kommunikationsförmåga
- Vara stresstålig

En annan roll som är tätt sammanknippad med kravhantering är en intressent. En intressent är en grupp eller individ som påverkas av eller på något sätt är ansvarig för uppdragets resultat. Intressenter är individer och organisationer som är aktivt involverade i projektet, eller vilkas intressen kan påverkas som ett resultat av projekts genomförande eller avslut. De kan vara antingen fysiska personer, juridiska personer eller teoretisk personer.

Det förekommer ofta intressekonflikter mellan intressenter vilket kan resultera i motstridiga krav. Problemet med motstridiga krav måste hanteras under kravanalysen.

Det kan finnas många kategorier av intressenter inklusive:

- Kunder
- Slutanvändare
- Projektledare
- Systemutvecklare eller -förvaltare
- Kunder i organisationer som kommer använda systemet
- Externa organ eller enheter (t.ex. tillsynsmyndigheter)
- Områdesexperter, t.ex. SME (Subject Matter Expert)
- Leverantörer av relaterade lösningar
- Förvaltningsorganisationer

Identifiering av alla intressenter är nödvändigt för att få förståelse för alla perspektiv rörande den planerade lösningen. Saknade intressenter kan resultera i saknade krav eller viktiga begränsningar som kan påverka omfattningen eller utformningen av lösningen.

För kurshållare: beskriv en typisk intressent, dess perspektiv på projektet och dess leverabler samt huvudmål och kommunikationsmönster (t.ex. produktägare/sponsor, projektledare, kund, kravhanterare, agilt utvecklingsteam, Scrum Master)

© SQEB – Software Quality Engineering Board

4 Kravledning	160 minuter

Begrepp

Baseline, Granskning, Horisontell spårbarhet, Konfigurationshantering, Konfigurationsobjekt, Kvalitetssäkring, Kvalitetskontroll, Mätetal, Produktrisk, Projektrisk, Påverkansanalys, Risk, Riskanalys, Riskhantering, Riskidentifiering, Riskhanteringsplan, Riskminskning, Spårbarhet, Vertikal spårbarhet, Ändring, Ändringsbegäran, Ändringshantering, Ändringsråd

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska inneha efter att slutförandet av respektive avsnitt.

4.2 Projekt- och riskhantering

LO-4.2.1	Förklara varför kravhantering är viktigt i projekt och ge lämpliga exempel (K2)
LO-4.2.2	Kunna återge de fel som kan uppstå i kravhantering (K1)
LO-4.2.3	Skilja mellan typiska projekt- och produktrisker relaterade till krav (K2)
LO-4.2.4	Med hjälp av exempel beskriva hur riskanalys och riskhantering kan användas för kravhantering (inklusive kravplanering) (K2)

4.3 Spårbarhet hos krav

LO-4.3.1	Kunna återge syftet med spårbarhet (K2)
LO-4.3.2	Känn igen olika typer av spårbarhet (K1)

4.4 Konfigurations- och ändringshantering

LO-4.4.1	Redogöra för vad som kännetecknar konfigurationshantering och ändringshantering samt ändringsrådets roll (K2)
LO-4.4.2	Förklara termerna: konfigurationsobjekt, defekt och ändringsbegäran (K2)

© SQEB – Software Quality Engineering Board

4.5 Kvalitetssäkring

LO-4.5.1	Kunna återge de faktorer som påverkar kvaliteten på kravhanteringsprocessen och dess produkter (K1)
LO-4.5.2	Förklara hur kravhanteringsprodukter stödjer test (K2)
LO-4.5.3	Förklara hur mätetal kan användas för att utvärdera och förbättra kvaliteten på kravhanteringsprocessen och dess leverabler (K2)

© SQEB – Software Quality Engineering Board

4.1 Introduktion till kravledning	10 minuter

Kravledning är huvudsakligen en uppsättning aktiviteter som behövs för att säkerställa att kravutvecklingsprocessen genomförs på rätt sätt under produktens livscykel. Även konfigurations- och ändringshantering är, precis som förvaltning av lösningen, kravledningsaktiviteter.

Kravledning hanterar vanligtvis följande aktiviteter:

- Spårbarhet hos krav
- Konfigurations- och ändringshantering
- Kvalitetssäkring

© SQEB – Software Quality Engineering Board

4.2 Projekt- och riskhantering	40 minuter

Huvudorsaker varför projekt misslyckas är relaterade till krav. Att försumma kravhantering kan resultera i att krav blir oprecisa, motsägelsefulla, inte uppfyller sina kriterier och inte tillfredsställer intressentens behov. Noggrann och strukturerad kravhantering är en nödvändig del av varje projekt.

För att minimera projektriskerna ska kravhantering bidra inom följande områden:

- Projektstart – kravhantering stödjer identifiering av kunder, intressenter, mål och visioner, behov och förväntningar rörande lösningen på problemet och stödjer upprättande av högnivåkrav.
- Avtals- och kontraktsförhandlingar – kravhantering stödjer utvärdering av kundens krav och fastställande av den inledande omfattningen, nödvändiga resurser och fastställande av kostnaden för utveckling (d.v.s. implementering av kraven)
- Projektdefinition – kravhantering inkluderar definitionen av roller, uppgifter, aktiviteter och ytterligare processer (t.ex. ändringshantering). Den bidrar också till den inledande arkitekturella designen och testprocessen.
- Projektgenomförande – i detta steg tillhandahåller kravhantering en grund för kravutveckling och verifiering och validering av krav (d.v.s. test). Den bör också övervara ändringar och begära granskningsplaner och justera dessa till den aktuella omfattningen av lösningen när kraven ändras.

Ineffektiv eller dålig kravhantering ökar projektriskerna genom att tillåta fel att fortplanta sig till efterföljande steg i utvecklingen av lösningen. Därför bör både praxis och vanliga risker och fel tas med i beräkningen när kravhanteringsprocessen planeras. De vanligaste problemen kopplade till kravhantering är följande:

- Otydliga verksamhetskrav och brist på verksamhetsmål som lösningen ska uppfylla
- Ändrade krav, ofta som ett resultat av otydliga projektmål och brist på en tydlig definition av kundens verksamhetsområde. Kravändringar ses inte som ett problem i agila och iterativa angreppsätt.
- Otydliga ansvarsområden (på både kundens och leverantörens sida)
- Skillnader (gap) mellan intressenterna förväntningar och det projektet levererar
- Otillräckligt intressentengagemang
- Brist på spårbarhet, vilket ofta resulterar i oprecisa estimat rörande kravändringars påverkan på andra områden hos produkten under utvecklingen
- Oprecisa estimat rörande kostnader eller omfattning, och/eller projektdefinition med milstolpar som är orealistiska (ofta som ett resultat av otydliga krav)

© SQEB – Software Quality Engineering Board

Potentialen för problemen ovan kan uppfattas som en risk. För att hantera dessa risker – och även andra risker – är en riskhanteringsprocess nödvändig.

Effektiv riskhantering är nyckeln till att minska projekt- och produktrisker. Identifiering, lämplig analys samt planering av adekvata riskåtgärder bidrar till att minimera chansen att en risk och dess medförande konsekvenser inträffar.

Risk är inte enbart kopplat till krav. Risk kan definieras som sannolikheten att en händelse eller situation uppstår och resulterar i oönskade konsekvenser eller potentiella problem. Risk kan uttryckas i nivåer, där risknivån bestäms av sannolikheten att den oönskade händelsen inträffar och dess konsekvenser (skadan som uppstår på grund av den händelsen) [ISTQB].

Det finns två huvudtyper av risker: produktrisker och projektrisker.

Projektrisker är de risker som rör projektets förmåga att leverera sina mål, såsom:

- Organisatoriska faktorer (t.ex. brist på kunskap, utbildning eller resurser, politiska frågor, problem med att få intressenter att uttrycka sina behov och förväntningar rörande den planerade lösningen, eller fel attityd eller förväntningar rörande kravhantering)
- Tekniska frågor (t.ex. problem med att definiera rätt krav eller tekniska/arkitekturella lösningar, till vilken utsträckning kraven inte kan uppfyllas givet befintliga begränsningar, eller dålig kvalitet på designen, koden, konfigurationsdata, test, krav- eller lösningsspecifikationer)
- Leverantörsfrågor (t.ex. tredjepartskomponenter som inte levereras i tid, eller kontraktsfrågor)
- Verksamhetsrisker på grund av bristande kvalitet (t.ex. risken att förlora kunder om den levererade lösningen befinns vara otillförlitlig eller ineffektiv)

Produktrisker är potentiella felområden (negativa framtida händelser eller faror) i programvaran eller systemet. Detta är risker rörande produktens kvalitet. Dessa inkluderar:

- Större risk för fel i den levererade produkten (programvara eller system som inte kan utföra en efterfrågad funktion inom specificerade ramar)
- Dålig kvalitet på lösningsdokumentationen (t.ex. användarmanualer, installationsguider)
- Risken att mjukvaran/hårdvaran kan orsaka skada för en individ eller företag
- Dåliga produktegenskaper (t.ex. funktionalitet, tillförlitlighet, användbarhet, prestanda)
- Dålig dataintegritet och -kvalitet (t.ex. problem med migrering, konvertering och överföring av data, överträdelse av datastandarder)
- En produkt som inte utför sina avsedda funktioner och inte uppfyller intressenternas behov

Riskhanteringsprocessen stödjer identifiering av potentiella faktorer som kan ha en negativ effekt på projektets genomförande och förberedelse av lämpliga åtgärder för att hantera risken om den inträffar.

Riskhantering består av följande aktiviteter [ISTQB]:

- Riskidentifiering

© SQEB – Software Quality Engineering Board

- Riskanalys (bedömning)
- Riskminskning

När risker ska identifieras är det viktigt att undersöka hela nätverket av intressenter eftersom olika intressentgrupper eller individuella intressenter kan upptäcka olika risker. T.ex. kan en sponsor fokusera mer på det värde lösningen kommer att tillhandahålla, medan den viktigaste risken för en kravhanterare kan vara bristande kommunikation med kundens verksamhetsrepresentanter.

Riskanalys omfattar fastställande av risknivåer – sannolikhet och möjlig konsekvens. Bägge attribut kan uttryckas i nivåer, eller – mer exakt – i en form av numeriska estimat (t.ex. sannolikheten att en oönskad händelse inträffar kan uttryckas som ett procenttal, medan konsekvensen kan uttryckas i termer av ekonomisk förlust).

Riskminskning planerar för lämpliga reaktioner på de risker som identifierats som mest centrala baserat på resultaten från riskanalysen. Grundläggande tekniker för att minska risker kan delas in i fyra huvudkategorier:

- Undvika risk
- Minska risk
- Dela risk
- Acceptera risk

När man arbetar med riskhanteringsprocessen är det viktigt att etablera och underhålla en riskhanteringsplan. Denna plan bör skapas innan projektplanen och sedan underhållas löpande (t.ex. efter varje iteration eller milstolpe). Riskhanteringsplanen bör tillhandahålla effektiva säkerhetskontroller för att hantera riskerna och ska även innehålla ett schema för implementation av kontroller samt en lista över ansvariga personer.

Riskhanteringsplanen innehåller vanligen:

- Lista över risker (klassificerade enligt typ)
- Ägare
- Sannolikheten för att respektive risk inträffar
- Allvarlighetsgrad för respektive risk (inklusive kostnad, om tillämpligt)
- Strategier för att minska varje risk (inklusive den person/grupp som är ansvarig för att utföra riskminskningsaktiviteterna)
- Spårbarhet mellan risker och krav och andra projektartefakter
- Riskbedömningsmatris

För kurshållare: ge exempel på risker kopplade till krav och exempel på aktiviteter för att hantera riskerna.

© SQEB – Software Quality Engineering Board

4.3 Kravspårbarhet	40 minuter

Krav är inte frysta, utan fortsätter att utvecklas under projektets och produktens livscykel. Följande är exempel på anledningar till fortsatt utveckling och föreslagna ändringar av kraven:

- Nya insikter eller kundbehov (t.ex. som ett resultat av nya regler, förordningar, nya produkter)
- Fortsatt arbete (t.ex. detaljering av redan definierade högnivåkrav, nästa projektfas, förbättring och optimering av redan implementerade features)
- Nya samband inom projektet (t.ex. integration med nya system, nya accesskanaler såsom internet eller mobila kanaler)

Beroende på graden av kravanalys och/eller implementation kommer kravet att befinna sig i olika tillstånd. Ett kravs livscykel uttrycks ofta genom olika status, t.ex.

- Nytt (föreslagen)
- Godkänt
- Motstridigt
- Implementerat
- Modifierat
- Borttaget
- Driftsatt

Olika processer (och status) för krav kan användas beroende på angreppsätt och organisation. I många fall är processerna för krav, ändringsbegäran och avvikelser likartade och hanteras med hjälp av samma verktyg.

Kravspårbarhet innebär en kontroll av att alla viktiga steg i utvecklingsprocessen har genomförts och bör implementeras dubbelriktat för alla artefakter (till exempel, från krav till designartefakter och från designartefakter till krav). Spårbarhet är också grunden för test, verifiering och validering.

Följande är generella mål för spårbarhet:

- Påverkansanalys
- Täckningsanalys
- Bevis på implementering
- Kravanvändning (kravspårning som ett bevis på att kraven används och hur de används)
- Återanvändning av krav

För att säkerställa bra spårbarhet är det viktigt att kraven har en unik märkning.

© SQEB – Software Quality Engineering Board

Det finns två grundtyper av spårbarhet:

- Horisontell spårbarhet
    - Visar på beroenden mellan olika typer av artefakter på samma abstraktionsnivå
- Vertikal spårbarhet
    - Visar på beroenden mellan artefakter på olika abstraktionsnivåer

I kravsammanhang används horisontell spårbarhet för att påvisa kopplingar mellan olika typer av krav (till exempel, horisontell spårning mellan ett funktionellt programvarukrav och ett GUI-krav som stödjer den specifika funktionen). Vertikal spårbarhet används ofta för att hantera kravtäckning på olika abstraktionsnivåer (till exempel, vertikal spårning mellan verksamhetskrav och lösningskrav).

Båda typerna av spårbarhet kan användas för att stödja påverkansanalys.

Enligt ISO/IEC/IEEE 29148–2011 (tidigare IEEE 830), kan spårning i statiska mallar implementeras genom:

- Textuella referenser i krav till identifierare i andra artefakter
- Textuella referenser i attribut
- Hyperlänkar
- Matriser

För kurshållare: förklara innebörden i spårbarhet och tillhandahåll exempel på tillämpning.

© SQEB – Software Quality Engineering Board

4.4 Konfigurations- och ändringshantering	30 minuter

För att säkerställa fullgod kravhantering måste en konfigurationshanteringsprocess vara implementerad. I många fall är kraven inte stabila och de efterföljande ändringarna kan påverka andra relaterade projektartefakter.

Syftet med konfigurationshantering är att etablera och underhålla integriteten hos produkterna (komponenter, data och dokumentation) och programvaruartefakterna genom projektets och produktens livscykel.

Konfigurationshantering är en disciplin som använder tekniska och administrativa verktyg och tekniker för att identifiera och dokumentera de funktionella och fysiska egenskaperna hos ett konfigurationsobjekt, kontrollera förändringar hos dessa egenskaper och registrera och rapportera bearbetnings- och implementationsstatus, och slutligen – verifiera överensstämmelse med specificerade krav [IEEE 610].

Ett konfigurationsobjekt är en artefakt, ett dokument eller en produkt (hårdvara och/eller mjukvara) som har ett syfte för slutanvändaren och behandlas som en unik enhet i konfigurationshanteringsprocessen [IEEE 610].

Vanliga konfigurationsobjekt som används inom kravhantering inkluderar:

- Individuella krav
- Kravspecifikationer
- Modeller

Konfigurationshantering säkerställer att kravhanteringens alla arbetsprodukter är identifierade, versionshanterade, har sina ändringar spårbara är länkade till varandra och länkade till andra projektobjekt (t.ex. utvecklings- och testartefakter) så att spårbarhet kan upprätthållas genom hela produktionsprocessen.

När projektet planeras ska procedurer och infrastruktur (verktyg) för konfigurationshantering väljas, dokumenteras och implementeras. Detta med anledning av att konfigurationsobjekt ska definieras och kunna ändringshanteras så tidigt som möjligt.

För kurshållare: ge exempel på konfigurationshanteringsaktiviteter för olika arbetsprodukter i kravhanteringsprocessen.

En viktig aspekt kopplat till konfiguration är ändringar. Ändringar av kraven kan efterfrågas när som helst under projektets genomförande och efter driftsättning av den färdigställda produkten i driftmiljön.

Källan till en ändring kan vara följande:

- Utvidgning eller förändring av existerande funktionalitet

© SQEB – Software Quality Engineering Board

- Begäran om ny funktionalitet
- Fel funna i produkten eller dokumentationen vilket resulterar i ett behov av ändring
- Ändringar som ett resultat av externa faktorer (t.ex. organisatoriska förändringar, förändringar i förordningar)

Ändringar är vanligt förekommande och det är viktigt att planera för ändringarna när det gäller process och tid.

Det är viktigt att skilja mellan en ändring och ett fel. Ett fel är en avvikelse från systemets förväntade tillstånd. En ändring är en modifiering eller ett tillägg av features, krav eller funktioner.

Ändringar hanteras via ändringshanteringsprocessen, vilken består i att efterfråga ändringen, bedöma möjligheten att genomföra den, planera, implementera och utvärdera förändringar av programvaran eller systemet, dokumentation eller andra projektprodukter. Syftet med ändringhantering är att stödja ändringsprocessen och säkerställa spårbarhet hos ändringar.

Ändringhanteringsprocessen omfattar vanligen följande aktiviteter:

1. Identifiera en potentiell ändring
2. Efterfråga ny eller förändrad funktionalitet (inkomma med en ändringsförfrågan)
3. Analysera ändringsförfrågan (görs vanligen av ett ändringsråd och innebär även påverkansanalys av ändringen)
4. Utvärdera ändringen (inklusive bedömning av risken, kostnaden och arbetsinsatsen som ändringen kräver som underlag för att kunna fatta ett beslut huruvida ändringen ska implementeras eller inte)
5. Planera ändringen (om den har godkänts för implementering)
6. Implementera ändringen
7. Granska och stänga ändringen
8. Leverera ändringen till test och produktionsmiljöer

En ändring bör inkomma som en formell ändringsbegäran (ofta Change Request (CR) eller Request for Change (RFC)). Detta dokument inkluderar följande information:

- Namnet på personen/avdelningen eller annan enhet som efterfrågar ändringen
- Datum för inkommande
- Anledning till ändringen
- En beskrivning av den efterfrågade lösningen
- Planerat (önskat) implementationsdatum för ändringen (om tillämpligt)
- Planerad (eller möjlig) budget för ändringen (om tillämpligt)

© SQEB – Software Quality Engineering Board

Ändringar kontrolleras och beslutas ofta av ett ändringsråd (Change Control Board (CCB) eller Configuration Control Board). Ett ändringsråd är en kommitté som fattar beslut rörande huruvida föreslagna ändringar ska implementeras eller inte grundat på tillgänglig information (såsom risken relaterad till ändringen, dess påverkan och vilken insats som krävs för implementering). Ändringsrådet består av projektintressenter. I agila utvecklingsmodeller utförs rollen som CCB vanligen av utvecklingsteamet tillsammans med produktägaren och kundrepresentanter.

Att införa ett ändringsråd ger stöd för att på ett kontrollerat sätt att granska, bedöma och implementera en ändring av en produkt eller tjänst.

Ändringsrådet består vanligen av följande roller:

- Projektledning
- Kravhanterare
- Utvecklingsledning
- Kvalitetssäkring (kvalitetsledning, testledning)
- Verksamhetsledning
- Kund och/eller användarrepresentanter

Beroende på komplexitet och påverkan kan en ändring ha olika effekt på systemet. Små ändringar kan kräva mindre modifieringar medan komplexa ändringar radikalt kan påverka systemets logik. Varje ändring bör analyseras noggrant för att identifiera relaterade risker och bedöma värdet av modifiering i förhållande till förutsagda risker.

När man ska besluta om förändringar är det viktigt att analysera den påverkan ändringen kan ha på projektet. Detta görs vanligen genom påverkansanalys. Påverkansanalys använder spårbarhet för att kontrollera vilka artefakter som kommer att behöva ändras eller åtminstone bör kontrolleras när en specifik ändring införs.

Ändringar i krav kan ha olika effekter på projektet. De vanligast förekommande effekterna är följande:

- Arbete som ett resultat av ändringen (beroende på fas i projektet):
    - Uppdatera analys- och designartefakter (t.ex. krav- eller lösningsspecifikationer)
    - Uppdatera teknisk dokumentation eller användardokumentation
    - Ändra teststrategin, testplaner och testfall
    - Uppdatera utbildningsbehov/-planer
    - Utöka/minska programmeringsomfattningen
    - Ändra testförberedelser och omfattning av genomförandet
- Förändringar av tidsplan, budget och resurser

Implementerade ändringar bör verifieras innan driftsättning i test-/produktionsmiljöer.

© SQEB – Software Quality Engineering Board

4.5 Kvalitetssäkring — 40 minuter

När kravhanteringsprocessen definieras är det också nödvändigt att definiera nödvändiga kvalitetssäkringsaktiviteter som ska användas för att säkerställa att de olika kravhanteringsprocesserna och deras produkter är av god kvalitet.

Kvalitetssäkring definieras som "alla planerade och systematiska aktiviteter som implementeras inom kvalitetssystemet, och demonstreras vid behov, för att tillhandahålla tillräckligt förtroende att en enhet kommer att uppfylla kvalitetskrav" [ISO 9000]. Dessa aktiviteter inkluderar praktiska tekniker och aktiviteter som används för att uppfylla kvalitetskraven.

För att uppnå den efterfrågade kvalitetsnivån behövs även kvalitetsstyrning. Huvudmålen med kvalitetsstyrning är att styra och övervaka kvaliteten hos produkter eller tjänster genom användande av praktiska metoder så att de möter specificerade kvalitetsstandarder. Dessa praktiska metoder som ingår i kravhantering inkluderar projekthantering, riskhantering, ändringshantering, verifiering och validering, granskning samt konfigurationshantering och kravspårning.

I kravhanteringssammanhang kan kvalitetsstyrning även fokusera på verifiering att den producerade kravhanteringsdokumentationen uppfyller relevanta kvalitetskriterier.

När ett projekts kvalitetssäkringsaktiviteter planeras är det viktigt att komma ihåg att det finns vissa faktorer som kan påverka den önskade kvalitetsnivån hos produkten. Exempel på sådana faktorer är:

- Kundens kvalitetskriterier eller förväntningar kopplade till kvalitetssäkringsprocessen
- Vilken typ av produkt som produceras (t.ex. produktens komplexitet eller målgruppen för produkten – en produkt som är avsedd för en liten målgrupp kan ha lägre nivå på vissa kvalitetsattribut än en produkt för en bred marknad)
- Miljön som produkten produceras i (t.ex. kan den tekniska miljön begränsa möjligheten att uppnå den önskade nivån av kvalitetsattribut)
- Bransch (t.ex. komplexiteten i verksamhetsbranschen, nivån av nyskapande, ändringsfrekvensen hos verksamheten)
- Legala, säkerhetsmässiga och miljömässiga faktorer (t.ex. förordningar kan resultera i att en högre nivå av kvalitet krävs)
- Tids- och kostnadstryck kan minska möjligheten att genomföra kravhanteringsprocesser

För att säkerställa den önskade kvalitetsnivån bör verifierings- och valideringsaktiviteter planeras och genomföras från projektets början (se avsnitt 5.5 Validering och verifiering av krav).

Följande verktyg och tekniker kan användas för kvalitetssäkring och kvalitetskontroll av krav:

- Standarder och mallar

© SQEB – Software Quality Engineering Board

- Spårbarhet
- Olika typer av granskningar
- Prototyparbete
- Iakttagande av kvalitetskriterier för krav och specifikationer

Kvalitetssäkring kan också stödjas genom att säkerställa kravens testbarhet. Testbara (verifierbara) krav kan kontrolleras att de implementerats korrekt och enligt intressentens behov. Testbarhet hos krav stöds av ett projekts acceptanskriterier (se avsnitt 5.5 Validering och verifiering av krav).

Kvaliteten hos en krav-/lösningsspecifikation kan förbättras genom att inkludera följande delar:

- Dokumentets syfte, omfattning, definitioner och ordlista
- Målet för olika nivåer (t.ex. en specifikation med högnivå-krav har andra mål än en detaljerad funktionsspecifikation)
- Begränsningar rörande design och implementation
- Prioritet eller graderingar rörande kraven
- Tydliga definitioner av vad ett system ska göra snarare än hur det ska göras
- Dokumentation för all applicerbar lagstiftning, alla antaganden, verksamhetsregler och beroenden
- Svåra, abstrakta texter ersatta med diagram där så är möjligt
- Tydligt specificerade användare och behörigheter
- Strukturerad presentation
- Enkelt, tydligt, precist och entydigt språk

Som framgår i avsnitt 3.2 Generisk kravhanteringsprocess, är krav den grundläggande informationen för indata till processen att utveckla och testa systemet. Väldefinierade krav minskar risken för problem (fel) i projekt och i synnerhet produkter eftersom de stödjer noggrann testning. Stabila krav ökar möjligheterna att möta deadlines.

Kravhantering har en nära koppling till test. Bra testfall kräver bra, testbara krav. Det är därför väldigt viktigt att involvera testare under utvecklingen av specifikationen. Kravhanteringsprodukterna stödjer testning genom att tillhandahålla basen för test. Krav och dess specifikationer benämns ibland testbas.

Det är viktigt att komma ihåg att kraven bör valideras genom statiska tester (granskningar) i vilka testare kan delta. Godkännande av testledare kan krävas, särskilt när kraven har visat sig vara otestbara. Testare kan hjälpa till att förbättra kravens kvalitet genom att identifiera svagheter och möjliga fel. Testa bör också delta i granskningen av kraven för att säkerställa testbarheten.

Att säkerställa och kontrollera kvalitet kräver vanligtvis mätetal. Ett mätetal är en mätskala och metoden som används för att mäta [ISO 14598]. Mätetal gör det möjligt att göra kvantifierbara uttalanden rörande projektets status och kvalitet. Det är viktigt att komma ihåg att resultatet av en

© SQEB – Software Quality Engineering Board

mätning (siffror insamlade under mätningen) alltid måste jämföras med referensdata (ett relevant mätetal).

Följande är exempel på mätetal som kan appliceras på krav:

- Projektkostnader
    - Antal krav och dess komplexitet
- Projektstabilitet
    - Antal ändrade krav
- Antal fel som hittats i krav eller lösningsspecifikationer
    - Typ av fel
    - Antal fel kopplat till olika typer (t.ex. logiska fel, konsistensfel, datafel)

Kvalitetsmätning hos krav är vanligtvis svårt eftersom det inte finns något enkelt sätt att uttrycka kvalitet. Därför mäts kvalitet vanligen genom checklistor som täcker in generella kvalitetsattribut. Följande är några frågor man kan använda sig av vid utvärdering av kravkvalitet:

- Är kravet korrekt?
- Är kravet entydigt?
- Är kravet mätbart?
- Är kravet spårbart?
- Går kravet att identifiera?
- Är kravet testbart?
- Är kravet designoberoende?

Även ändringsfrekvensen kan mätas som ett mått på kravkvalitet (detta gäller inte agila projekt). Ändringar görs för att tydliggöra otydliga krav och korrigera inkompletta eller inkonsistenta krav. Ju högre ändringsfrekvens hos en uppsättning krav, desto större projektrisk. Därför bör ändringsfrekvensen också mätas för att hantera risker inom ett projekt.

För kurshållare: ge exempel på mätetal som mäter kvaliteten hos kraven och kravhanteringsprocessen.

© SQEB – Software Quality Engineering Board

5 Kravutveckling	290 minuter

Begrepp

Acceptanskriterier, Analys av användningsfallspoäng, Analys av funktionspoäng (Function Points), Användningsfall, Användarberättelse, BPMN, Brainstorming, Delphi, Enkät, Formaliseringsgrader, Fältstudier, GUI-modellering, Identifiering baserat på befintlig dokumentation, Intervju, Konceptuell modell, Kontrakt, Kravinsamling, Kravmodell, Kravspecifikation, Kundlärande, Kundrepresentant, Lösningsmodell, Lösningsspecifikation, Modell, Mål, Planning Poker, Prioritering, Självregistrering, S.M.A.R.T., Specifikation, SysML, UML, Vision, Workshop, Återanvändning

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska inneha efter att slutförandet av respektive avsnitt.

5.2 Kravinsamling

LO-5.2.1	Förstå målen med kravinsamling (K2)
LO-5.2.2	Förklara de vanligaste källorna till krav och ge några exempel (K2)
LO-5.2.3	Förstå en typisk projektvisions innebörd för kravhantering (K2)
LO-5.2.4	Förstå intressentens roll vid kravinsamling och förklara hur intressenter kan identifieras (K2)
LO-5.2.5	Förklara tillämpningen av olika tekniker för kravinsamling och applicera dem på ett givet scenario (K3)

5.3 Kravanalys

LO-5.3.1	Förklara målet med och huvudaktiviteterna inom kravanalys (K2)
LO-5.3.2	Förklara skillnaden mellan krav och lösningar (K2)
LO-5.3.3	Förklara anledningen till estimering av en insats och ge exempel på olika angreppssätt för estimering (K2)
LO-5.3.4	Tillämpa proceduren för kravprioritering för ett givet scenario (K3)
LO-5.3.5	Förklara syftet och konsekvenser av kravacceptans (K2)
LO-5.3.6	Återge olika modeller som kan användas inom kravanalys (K1)

© SQEB – Software Quality Engineering Board

LO-5.3.7 Förstå tillämpningen och kunna skapa grundläggande analysmodeller (kontextanalys, funktionell nedbrytning, verksamhetsprocessmodeller) (K3)

LO-5.3.8 Förklara egenskaperna hos grundläggande UML-diagram (tillståndsmaskindiagram, användningsfallsdiagram, aktivitetsdiagram, och klassdiagram) och möjliga tillämpningar av dessa(K2)

5.4 Kravspecificering

LO-5.4.1 Förstå och förklara användningen av användningsfall och användarberättelser i utarbetandet av kravspecifikation och lösningsspecifikation (K2)

LO-5.4.2 Förstå och förklara syfte, egenskaper, innehåll och typiskt arbetssätt för att skapa en lösningsspecifikation (K2)

LO-5.4.3 Beskriva vanliga egenskaper hos olika formaliseringsgrader och förklara när en specifik grad av formalisering kan användas (K2)

5.5 Validering och verifiering av krav

LO-5.5.1 Summera tekniker för verifiering och validering som kan användas för att undvika fel i kraven (K2)

LO-5.5.2 Förstå användningen av acceptanskriterier (K2)

© SQEB – Software Quality Engineering Board

5.1 Introduktion till kravutveckling	10 minuter

Kravutvecklingsprocessen täcker, som tidigare definierat, följande aktiviteter:

- Kravinsamling
- Kravanalys
- Kravspecificering
- Validering och verifiering av krav

Dessa aktiviteter ska inte ses som enskilda faser utan är sammankopplade. Ofta finns det ett högt marknadstryck på kortare utvecklingscykler samtidigt som det finns risker med frekventa ändringar, uppgraderingar eller revideringar av lösningen under utvecklingen. Det är därför i det närmaste omöjligt att implementera kravutvecklingsprocessen som en linjär process där krav samlas in från intressenterna, analyseras, specificeras och lämnas över till utvecklingsteamet som sedan skapar en lösningsmodell och validerar kraven. I verkligheten itererar kraven vanligtvis mot en kvalitets- och detaljeringsnivå som är tillräcklig för att möjliggöra att beslut rörande design och upphandlingar kan fattas. I så gott som alla fall kommer kraven att ändras, vilket betyder att lämpliga åtgärder måste vidtas för att minska ändringens påverkan.

© SQEB – Software Quality Engineering Board

5.2 Kravinsamling	100 minuter

Det första steget i kravutvecklingsprocessen är vanligen kravinsamling. Huvudmålet med kravinsamling är att samla in krav från alla tänkbara intressenter – inte bara användare och sponsorer, utan även projektgruppen, marknaden och andra externa källor. I allmänhet är huvudsyftet med kravinsamling följande:

- Identifiera alla önskade funktioner, egenskaper, begränsningar och förväntningar
- Styra kraven mot projektvisionen
- Detaljera högnivåkrav och beskriva funktioner och tjänster tydligt
- Exkludera funktioner och features som kunden inte vill ha

Följande kan vara typiska kravkällor:

- Dokument
- System som redan används
- Intressenter

Dessa källor kan påverka de valda teknikerna för kravinsamling.

Kravinsamling handlar inte bara om att samla in intressenters behov genom att ställa frågor – mycket ofta behöver den insamlade informationen tolkas, analyseras, modelleras och valideras innan en komplett uppsättning krav på en lösning kan fastställas. Vilka insamlingstekniker och verktyg som används kan ibland styras av val av modelleringsteknik eller övergripande strategi för analys. Många modelleringstekniker innebär (förespråkar) även användande av särskilda insamlingstekniker.

Det första steget i kravinsamling är att komma fram till vilket problem det är som ska lösas. Detta innebär identifiering av intressenter och upprättande av (eller insikt om tidigare identifierade) verksamhetsmål på hög nivå. Intressenterna tillhandahåller krav och begränsningar. Det är därför väldigt viktigt att identifiera alla intressenter. Affärsmål stödjer upprätthållandet av visionen om vad som ska göras – krav som samlas in under kravinsamlingen bör stödja uppfyllandet av verksamhetsmålen. Att se till att kraven stämmer överens med verksamhetsmålen gör det också lättare att styra lösningens omfattning.

Inledande krav tillhandahålls av kunden. Kunden är en av nyckelintressenterna i projektet och dess behov måste tillfredsställas.

Kunden måste alltid vara involverad. Målet är att uppnå en ömsesidig förståelse mellan kund och leverantör. Leverantören bör alltid tänka sig själv in i kundens situation.

En av de viktigaste källorna till krav är ett kontrakt mellan leverantören och kunden. Överenskommelsen (kontraktet) ska formellt specificera och beskriva vad kunden vill ha. Det måste

© SQEB – Software Quality Engineering Board

säkerställas att kundens intressen hamnar i centrum (vilket innebär att leverantören inte ska tvinga på kunden den lösningen man själv föredrar utan ska analysera kunden behov och rekommendera en lösning som på bästa sätt uppfyller dessa behov).

Kontraktet måste vara i enlighet med tillgängliga resurser för att implementera lösningen och ska baseras på: estimat, deadlines, priser och projektplaner. Kravhantering tillhandahåller indata för sådana estimat.

Kontraktet bör innehålla:

- En kort beskrivning av den planerade lösningen
- Listan med prioriterade krav
- Acceptanskriteriet för varje krav
- Lista med produkter som ska tillhandahållas (t.ex. dokumentation, kod, fungerande programvara, tjänster, processer)
- Tidsplan med tidsramar för utveckling och leverans av produkten
- Andra behov och förväntningar så som vilken teknik som föredras, resurskrav, etc.

En annan viktig källa som uttrycker kundens primära behov är projektvisionen. För varje projekt måste en ny vision tas fram. En vision ska på en hög nivå definiera de mål som ska uppnås. Det är av avgörande betydelse att ha en tydlig definition i projektvision eftersom visionen utgör ett sätt att verifiera huruvida projektet uppnår det efterfrågade värdet eller inte.

En vision utgörs av mål som ska uppfyllas av ett specifikt projekt. Mål ska uttrycka projektets verksamhetssyfte och ska uppfylla definitionen S.M.A.R.T. för att möjliggöra kontroll om de är uppfyllda när projektet är klart. S.M.A.R.T. är en metod som används för att etablera mål och definiera deras kvalitetsmål. S.M.A.R.T. kräver att alla mål har följande egenskaper [G.T, Doran]:

- S – Specifikt
- M – Mätbart
- A – Uppnåeligt
- R – Relevant
- T – Tidssatt

När krav samlas in är det viktigt att säkerställa att kraven både kommer att uppfylla projektvisionen och definitionen av S.M.A.R.T.:a mål.

Följande processer kan hjälpa vid upprättande av visioner och mål:

1. Hitta källor till krav
2. Analysera den nuvarande situationen (utvärdering av målsättning)
3. Utvärdera den nuvarande situationen (hänsyn tas till subjektiva faktorer)
4. Dra slutsatser om vision/mål (baserat på beräkningar av kostnad/nytta)

© SQEB – Software Quality Engineering Board

För att kunna identifiera alla krav och behov måste alla intressenter på både kundsidan och leverantörssidan identifieras. Varje intressent eller grupp av intressenter kan skapa nya krav och påverka utformningen av den planerade lösningen. Om inte alla intressenter identifieras finns det en risk att några viktiga krav eller begränsningen fortsätter vara okända och inte tas med i beräkningen när lösningen designas. Saknade intressenter kan resultera i att komplexa ändringar av produkten efterfrågas i senare skeden av projektet eller efter att systemet driftsatts i produktionsmiljön.

Intressenter bör beskrivas i följande termer: namn, funktion, tillgänglighet, domän, mål och intressen.

Vissa intressenter kan ingå i intressegrupper (t.ex. alla verksamhetsintressenter). Intressegrupper bör sammanställas eftersom det möjliggör att deras krav kan hanteras mer effektivt.

Proceduren att identifiera och utvärdera intressenter innehåller följande aktiviteter:

- Identifiering av intressenter (görs genom analys av verksamhetsprocesser, fastställande av process- och produktägare, analys av organisatorisk struktur och marknad)
- Gruppering av intressenter och val av representanter för specifika grupper (om möjligt)
- Fastställande av kopplingar mellan intressenter
- Identifiering av potentiella konflikter
- Analys av konflikter, fastställande av dess källor och identifiering av möjligheter till nytta för bägge sidor (win-win opportunities)
- Identifiering av risk-minimerande intressenter för att kunna involvera dem i projektaktiviteter
- Identifiering av intressentens perspektiv

Om intressenter, verksamhetsmål och vision är kända är det möjligt att starta med detaljerad kravinsamling.

De vanligaste teknikerna för kravinsamling är följande:

- Enkäter
- Intervjuer
- Självloggning
- Insamling av information från kundrepresentant på plats
- Identifiering utifrån befintlig information
- Återanvändning (Återanvändning av specifikationer från ett annat projekt)
- Brainstorming
- Observation
- Kundlärande

© SQEB – Software Quality Engineering Board

- Workshops
- Användningsfall

Enkäter

Beskrivning

En enkät kan bestå av öppna eller slutna frågor. En öppen fråga innebär att respondenten formulerar sitt eget svar. En sluten fråga innebär att respondenten väljer ett svar från ett antal möjliga svarsalternativ. Dessa svarsalternativ ska vara ömsesidigt uteslutande.

Tillämpning

För att säkerställa eller detaljera kända krav; organisera kravinnehåll; välja alternativ för krav/lösning.

Nackdelar

Inte lämpligt för att samla in underförstådda krav.

Låg svarsfrekvens om respondenten inte är motiverad.

Kan lätt bli styrande, vilket kan förhindra identifiering av verkliga användarbehov.

Intervju

Beskrivning

En samtalsteknik där den som intervjuar ställer frågor till den som intervjuas för att samla in information om utpekade ämnen. Denna teknik är väldigt interaktiv och tillåter anpassning av ordningen för sedan tidigare förberedda frågor beroende på svar och situation. En bra intervju är svårare att genomföra än vad man kan tro beroende på normalt samtalsbeteende (t.ex. att avsluta meningarna åt sin samtalspartner), vilket kan leda till tolkningar kommer med i insamlad data. Intervjuaren bör ställa öppna frågor för att inhämta information och stängda frågor för att bekräfta eller förkasta specifika val.

Tillämpning
För att samla in information inom ett speciellt ämne; för att tydliggöra krav.

Nackdelar
Tidskrävande
Bristfälligt återskapande av resultat (svårigheter att få samma svar när man upprepar en intervju)

Självloggning

Beskrivning
Denna teknik innebär att intressenten (t.ex. en slutanvändare) dokumenterar de aktiviteter han eller hon gör för att utföra en specifik uppgift. Utöver att dokumentera aktiviteterna som de utförs idag beskriver användaren också ändringar, önskemål och behov.

© SQEB – Software Quality Engineering Board

Denna teknik sammanknippas ofta med demonstrationer eller granskningar av dokument.

Tillämpning

För att förstå komplexa procedurer eller processer, särskilt när kravhanteraren inte har möjlighet att se när användaren utför sina uppgifter.

Nackdelar

"Automatiserade" aktiviteter försummas lätt

Väldigt beroende av användarens motivation och erfarenhet

Kundrepresentant på plats

Beskrivning

Detta angreppssätt är en av de mest effektiva metoderna för kravinsamling (och -validering) genom att det möjliggör för kundrepresentanten att systematiskt övervaka progressen, bekräfta att designen är rätt och ge feedback och kompletterande information när det behövs.

Att ha kundrepresentanter på plats är en av huvudreglerna i agila metoder.

Tillämpning

Att samla in och hantera krav i agila metoder; att tillhandahålla användarorienterade krav som lätt kan accepteras.

Nackdelar

Hög kostnad för kunden

Kostnader för anpassningar

Kravidentifiering baserad på befintliga dokument

Beskrivning

Denna teknik kan användas när det finns befintlig dokumentation som kan hjälpa till att identifiera krav inom en organisation. Exempel på denna typ av dokumentation är processmodeller och -kartor, processbeskrivningar, organisationsscheman, produktspecifikationer, procedurer (t.ex. arbetsprocedurer), standarder och instruktioner, dokumentmallar, etc.

De krav som identifieras är grunden för vidare kravanalys och behöver detaljeras och utökas men andra relaterade och länkade krav.

Tillämpning

Att samla in krav för en lösning som skall täcka befintliga verksamhetsprocesser.

Nackdelar

Inte användbar när det enbart finns grundläggande dokument eller inga dokument alls inom en organisation.

© SQEB – Software Quality Engineering Board

Inte användbar när dokumentationen inte underhålls på rätt sätt (inte uppdateras).

Återanvändning (Återanvändning av ett visst projekts specifikationer)

Beskrivning

Återanvändning av specifikationen för ett visst projekt är möjlig när organisationen redan har färdigställt ett eller flera projekt som liknar det aktuella projektet. En kravspecifikation som skapats för ett (eller flera) tidigare projekt kan användas i ett annat projekt för att korta ledtiden för kravanalys och insatsen som krävs för att förbereda dokumentation. Detta kan göra det möjligt att komma igång med implementeringen tidigare.

I de flesta fall kan bara delar av existerande specifikationer användas i ett nytt projekt. Dokumentationen som ska återanvändas bör alltid kontrolleras så den överensstämmer med nuvarande behov och krav och anpassas därefter.

Tillämpning

För projekt som har som mål att tillhandahålla produkter som är anpassade efter specifika kunder. För att förkorta kravhanteringsarbetet i projekt som utvecklar liknande lösningar.

Nackdelar

Höga kostnader för det första projektet. Krav som är för detaljerade och resulterar i omfattande och kostsam ändringhantering.

Observation

Beskrivning

Observationer innebär att iaktta användarens aktiviteter och processer, och identifiera systemkravet utifrån dessa. Observationer på plats genomförs genom att iaktta när användarna arbetar och att dokumentera processer, arbetsuppgifter och resultat. I vissa fall utökas observationerna med intervjuer av användare rörande deras arbete och hur de utför sina uppgifter.

Tillämpning

För att samla in precisa krav genom att observera användarna i deras arbete. För att undvika problem som intressenter kan ha med att uttrycka sina behov. För att förbättra existerande verksamhetsprocesser eller system.

Nackdelar

Undantagsfall kan förbises. Inte applicerbart i vissa situationer (till exempel, av säkerhetskritiska eller juridiska anledningar)

Kundlärande

Beskrivning

© SQEB – Software Quality Engineering Board

Syftet med kundlärande är att samla in krav från en kund, särskilt när processerna och aktiviteterna som utförs av kundens personal inte är enkel att beskriva genom andra tekniker, som t.ex. intervjuer, eller att kunden har problem att beskriva sina krav.

Kundlärande är processen att lära sig om jobbet från kunden. Kunden, som vet bäst rörande hur ett specifikt jobb ska utföras, lär upp kravhanteraren – som en lärare och elev.

Tillämpning

För att förstå komplexa verksamhetsprocesser för att kunna föreslå den lämpligaste lösningen; för att överkomma de svårigheter som kundens anställda kan ha att tänka abstrakt och uttrycka sina arbetsuppgifter i ord.

Nackdelar

Dyrt och tidskrävande. Inte användbart i farliga miljöer.

Användningsfall

Beskrivning

Användningsfall låter kravhanteraren beskriva funktionaliteten hos lösningen från aktörens perspektiv. En användare kan vara antingen en slutanvändare eller ett externt system.

Användningsfallsdiagram anger lösningens gränser och visar vilka kopplingar som finns mellan externa system och aktörer.

Tillämpning

Att samla in och förklara krav från användarens perspektiv; för att tydliggöra och organisera funktionaliteten som efterfrågas hos lösningen.

Nackdelar

Inte tillämpligt i fall där lösningen till största delen utgörs av icke-funktionella krav.

Workshop

Beskrivning

En workshop är en typ av möte som fokuserar på ett specifikt ämne (i förväg avgränsat och meddelat till deltagarna), och involverar vanligen intressenter från olika områden och/eller domäner under kort, intensiv period.

En workshop involverar människor som har olika synvinkel på ett givet problem och hjälper till att fastställa och beskriva krav som kommer ur olika perspektiv.

Workshops är ett av huvudarbetssätten i agila metoder eftersom de involverar alla viktiga intressenter i framtagandet av produktbacklogen. Väl förberedda och genomföra workshops hjälper till att minska möjliga svårigheter så som brist på samsyn.

Tillämpning

© SQEB – Software Quality Engineering Board

För att identifiera krav i syfte att fastställa omfattningen på lösningen; för att upptäcka dolda krav (t.ex. krav som inte är direkt uttalade eller ens har förståtts av intressenten men som behövs för att uppfylla vissa av deras behov eller krav på högre nivå); för att utveckla eller detaljera krav i nyligen identifierade områden; för att prioritera krav eller uppnå konsensus när det gäller att uppnå överenskommelse om kraven (sign off); för att upptäcka och lösa potentiella konflikter mellan intressenters krav; för att granska resultat från specifika processer eller aktiviteter och lösa problem som kan ha uppstått.

Nackdelar

Kan vara svårt att med geografiskt spridda team

Kräver att alla personer som behöver delta i workshopen är tillgängliga

Konsensus är inte nödvändigtvis enkelt att uppnå under en workshop, och diskussioner kan fastna vid (mindre) problematiska frågor vilket kan göra processen långdragen och omotiverande för deltagare

Brainstorming

Beskrivning

Brainstorming är vanligt förekommande teknik för att fånga krav som är relaterade till områden rörande en organisations aktiviteter eller planerade systemfunktioner som är nya eller relativt okända. Det gör det möjligt att samla in idéer från olika intressenter under en kort tidsperiod och till låg kostnad. Under brainstorming-tillfället kommer deltagarna med idéer och koncept rörande ett givet problem.

Tillämpning

För att lösa kravkonflikter; för att definiera olika lösningsalternativ; för att lösa affärsproblem (t.ex., låg effektivitet hos en process).

Nackdelar

Svårt om deltagarna inte är motiverade

Svårt att tillämpa med geografiskt spridda team

För kurshållare: ge exempel på tillämpning av minst tre olika tekniker för kravinsamling.

För att uppnå goda resultat och undvika ”luckor” i kraven, används ofta en kombination av de insamlingstekniker som beskrivs ovan.

Som tidigare nämnts specificerar funktionella krav systemets funktioner som de upplevs av slutanvändaren. Funktionella krav beskriver också triggers i processen såsom användarstyrda händelser eller in-/utdata som startar verksamhetsprocessen.

När krav samlas in är det viktigt att inte bara fråga om funktioner, utan även om kvalitetsegenskaper. Icke-funktionella krav beskriver kvalitetsattribut hos ett helt system eller dess specifika komponenter eller funktioner. De kan begränsa lösningen, till exempel genom att ställa krav på

© SQEB – Software Quality Engineering Board

särskilda prestadaparametrar. Icke-funktionella krav är svåra att beskriva och ofta uttryckta på ett vagt sätt eller inte dokumenterade överhuvudtaget vilket gör dem svåra att testa. Därför bör särskild uppmärksamhet ägnas åt icke-funktionella krav på alla nivåer av kravhanteringsprocessen för att säkerställa att de uttrycks tydligt och mätbart.

Icke-funktionella krav kan beskriva olika aspekter av lösningens beteende (utförande). Enligt ISO/IEC 25000 (tidigare 9126) är följande särskilda typer av icke-funktionella krav:

- Tillförlitlighet
- Användbarhet
- Effektivitet
- Underhållbarhet
- Portabilitet

Icke-funktionella krav är särskilt viktiga eftersom de specificerar kriterier som kan användas för att bedöma systemets funktion och har därför stor påverkan på hur nöjd kunden med att använda produkten. Funktionella krav ska tillhandahålla funktioner; icke-funktionella krav avgör bland annat hur lätt och effektivt funktionerna kan användas.

För kurshållare: ge exempel på funktionella och icke-funktionella krav enligt ISO/IEC 25000.

När kraven väl är insamlade, ska de dokumenteras på lämpligt sätt för att underlätta framtida spårning och kravanalys.

Krav måste specificeras på ett tydligt och korrekt sätt. De ska vara mätbara för att kunna säkerställa att de är testbara och att implementationen av kraven kan testas på rätt sätt. Det är viktigt att komma ihåg att vardagligt språk har vissa begränsningar och nackdelar. Detta kan orsaka att kraven blir otydligt och tvetydigt beskrivna. Därför bör lämpliga standarder och mallar användas när det är tillämpligt. Standarder ger en gemensam förståelse och bra praxis rörande hur specifikationer bör skrivas när mallar begränsar det språk som kan användas.

Utöver standarder och mallar är också ordlistor ett viktigt verktyg för att underlätta kommunikation mellan olika intressenter och för att skapa viss kontroll över den otydlighet som vardagsspråk kan medföra.

Beskrivningen av ett krav måste uppfylla kvalitetskriterier (se 1.1.4 Kravkvalitet)

Beroende på abstraktionsnivån kan krav beskrivas med mer eller mindre detaljerat. I vissa utvecklingsmodeller kan verksamhetskrav beskrivas i form av användningsfall på hög nivå (till exempel, Rational Unified Process) eller användarberättelser (agila angreppsätt). I allmänhet bör den typiska strukturen hos ett verksamhetskrav täcka följande aspekter:

- Användaren – vem skulle vilja ha detta krav?
- Resultatet – vad är resultatet som intressenterna letar efter
- Objekt – vilket är objektet som kraven riktas mot?
- Kriterierna – vilka är de mätbara kriterierna?

© SQEB – Software Quality Engineering Board

Till exempel, kravet: ”Försäkringsombudet måste få information om alla varje ny produkt en dag innan produkten lanseras.” täcker de fyra delarna som listas ovan på följande sätt:

- Försäkringsombudet ← vem
- måste få information om ← vilket resultat
- varje ny produkt ← vilket objekt
- en dag innan produkten lanseras ← kvalificerare

Mer detaljerade krav (lösnings-/systemkrav) kan beskrivas i form av användningsfall, ofta tillsammans med scenarier eller som användarberättelser i agila angreppssätt. I allmänhet bör lösnings-/systemkrav täcka följande aspekter:

- Processer som påverkas – vilken verksamhetsprocess refererar kraven till? Denna del fokuserar vanligen på funktionalitet och fastställande av in- och utdata.
- Lösningsaktivitet – hur triggas aktiviteten? Är det en fristående systemaktivitet, eller en interaktion med användaren, eller ett gränssnitt till andra system eller processer?
- Juridiskt åtagande – vad är kravets juridiska åtagande rörande kravet? Beskrivs ofta med hjälp av vanliga nyckelord (bör, måste, etc.).
- Begränsningar i logik och tid – finns det några gränssättande villkor som är applicerbart för kravet?

För kurshållare: diskutera exempel på beskrivningar av verksamhetskrav och lösnings-/systemkrav

Identifierade och analyserade krav dokumenteras vanligen i form av en specifikation (se 5.4 Kravspecificering).

© SQEB – Software Quality Engineering Board

5.3 Kravanalys	80 minuter

Steget efter kravinsamling är kravanalys. Det huvudsakliga målet med analys är att skapa underlag för affärslösning som sedan implementerar kraven. Att skapa underlag för affärslösningen kräver vidareutveckling av kundens behov till system-/lösningskrav ända ner till produkt-/komponentkrav. Det innebär också att upptäcka och lösa inkonsistensproblem mellan krav, både på samma nivå och mellan olika nivåer, och att hitta gränser för lösningen som inkluderar hur den ska interagera med sin omgivning.

Den iterativa proceduren på hög nivå för kravanalys innehåller följande steg:

1. Analys av behoven
2. Beskrivning av lösningen
3. Estimering och prioritering av arbetsinsats
4. Kravöverenskommelse

Det första steget syftar till att analysera intressentens behov för att kunna föreslå en lösning som uppfyller dessa behov. I detta skede inkluderas ofta verifierings- och valideringsaktiviteter i kravanalysen för att säkerställa att kraven uppfattats korrekt och accepteras av intressenterna. En viktig del av kravanalysen är att kontrollera kravens tydlighet och konsekvens och att lösa eventuella inkonsistensproblem.

Beskrivningen av lösningen görs vanligen med hjälp av modeller. I allmänhet finns det tre grundtyper av analysmodeller: kravmodeller, lösningsmodeller och konceptuella modeller. Modeller utvecklas genom lämpliga analysmetoder.

Nästa steg i kravanalysen är estimering av arbetsinsats samt prioritering av kraven.

Estimering av arbetsinsats sammankopplar kravanalys med projektledning. Tidiga estimat av arbetsinsats grundar sig på högnivåkrav och används för att estimera projektets kostnad. När kraven vidareutvecklas används de inledande kostnadsestimaten för att styra projektet och vid eventuella ändringar för att estimera insatsen och kostnaden som dessa ändringar medför.

Estimat fokuserar vanligen på följande:

- Kostnad
- Tid
- Krav (typ, omfattning och innehåll)

Projektets kostnader beror på olika faktorer, inklusive:

- Projekttyp
- Processmognad
- Angreppsätt för design och test, metoder och verktyg
- Vilken teknik som används

© SQEB – Software Quality Engineering Board

- Komplexiteten hos den planerade lösningen
- Kvalitetsmål (till exempel, den önskade kvalitetsnivån på produkten)
- Teamets kompetens
- Teamets sammansättning
- Erfarenheter

Exaktheten hos estimaten beror på projektets progress och mognad.

Estimering kan i allmänhet utföras genom analogisk slutledning och algoritmiska procedurer. Båda angreppssätten kräver att estimeringsprocedurerna baseras på historisk data och gränssättande villkor.

Analogisk slutledning estimerar insats baserat på jämförelse med liknande projekt. Estimaten baseras på erfarenhet, inte på matematiska formler. Med denna teknik jämförs det aktuella projektet med tidigare projekt. Jämförelsen kan inkludera:

- Antal och komplexitet hos kraven
- Stabilitet i verksamhetsprocesserna
- Lösningens omfattning
- Använd teknik
- Medarbetarnas egenskaper (t.ex. kompetens, erfarenhet)

Följande är exempel på analogiska procedurer:

- Planning Poker
- Delphi-metoden

Ett exempel på en analogisk procedur kan hittas i agila projekt. I sådana projekt, särskilt när de hanteras med hjälp av Scrum, finns en estimeringsteknik som kallas Planning Game eller Planning Poker. Denna metod används för att uppnå konsensus inom teamet. Teamets förmåga mäts sedan genom någonting som kallas Burn Down Rate. Retrospektiv-möten genomförs efter varje sprint där de estimerade poängen jämförs med den faktiska tiden som gick åt. Detta förbättrar teamets förmåga att estimera.

Delphi-metoden är ett annat exempel på analogisk teknik. Det är en strukturerad kommunikationsteknik som används för att genomföra interaktiv prognostisering. En “expertpanel” sätts samman [Linstone75] och experterna ombeds besvara frågeformulär i ett antal omgångar. Efter varje omgång görs en anonym summering av de prognoser experterna gjort tillsammans med skälen för deras bedömningar. Experterna reviderar sedan sina tidigare estimat med hänsyn taget till de övriga panelmedlemmarnas svar.

Man gör antagandet att svarens spridning genom denna process kommer att minska och gruppen kommer att sträva mot ”rätt” svar. Processen stoppas vid på förhand bestämda kriterier. Medelvärdet av den slutliga rundan utgör resultatet.

© SQEB – Software Quality Engineering Board

Det andra angreppssättet för att estimera är en algoritmisk procedur. I detta fall beräknas insatsen med olika parametrar som grund. Parametrar kan beskriva produkten (volym, livslängd), gränssättande villkor (prestanda), etc. Exempel på algoritmiska procedurer är:

- Analys av funktionspoäng (Function Points)
- Analys av användningsfallspoäng (Use Case Points)

Analys av funktionspoäng kräver att man räknar så kallade funktionspoäng hos den tänkta lösningen. En funktionspoäng är en mätenhet för att uttrycka mängden affärsfunktionalitet som ett informationssystem ger en användare. Kostnaden för ett enskilt funktionspoäng beräknas utifrån tidigare projekt. Identifierade funktioner viktas utifrån komplexitetsnivåer; antal poäng som tilldelas varje nivå beror på vilken typ av funktionspoäng det är.

Analys av användningsfallspoäng påminner om analys av funktionspoäng men kräver utöver analys av andra objekt även analys av antal och komplexitet hos användningsfall och aktörer för den planerade lösningen.

En annan viktig aktivitet i kravanalysarbetet är prioritering. Prioritering bestämmer ett kravs relativa betydelse för implementationen. Detta möjliggör för teamet att färdigställa de viktigaste kraven först. Prioritering stödjer inkrementell utveckling eftersom krav kan grupperas utifrån prioritet och krav med högst prioritet kan planeras in för implementation i de tidigaste iterationerna.

Proceduren för att bestämma kravens prioritet innehåller följande aktiviteter:

1. Gruppera krav – vilket inkluderar att identifiera krav som påverkar eller är beroende av andra krav (t.ex. krav som tillsammans utgör en komplex funktionalitet).
2. Analys av kraven – analys av de grupperade kraven av alla inblandade intressenter för att kunna komma överens om nivån av viktighet och fastställa kravens prioritet. Påverkansanalys görs ofta för att stödja analysen.
3. Framtagande av projektplan för kraven – etablera en plan där kraven med hög prioritet ska utvecklas först samt utse ansvariga för implementeringen av kraven.
4. Planera testomgångar - designa testfall utifrån kravprioritet.

Ett vanligt angreppssätt för prioritering är att gruppera krav i prioritetskategorier. Ofta används en gradering i tre nivåer (t.ex. hög, medel, låg). Eftersom sådana skalor är subjektiva och oprecisa måste alla inblandade intressenter vara överens om betydelsen av varje nivå i skalan de använder. Definitionen av prioriteten ska tydligt framgå och vara ett nyckelattribut hos varje krav.

För kurshållare: förklara procedur för kravprioritering med några exempel. Förklara syftet med kravprioritering för leverans-/iterationsplanering.

Kravöverenskommelse (även kallat kravacceptans eller signoff) är nästa del av kravanalysen. Det är en formell överenskommelse om att innehållet om omfattningen av kraven är rätt och komplett. Det är viktigt att säkerställa att kraven har accepterats vid olika faser vid lösningsutvecklingen eftersom formella överenskommelser är grunden för vidare aktiviteter. Överenskommelse om högnivåkrav (affärskrav/verksamhetskrav) ska erhållas innan projektet startar. Överenskommelse

© SQEB – Software Quality Engineering Board

om de detaljerade kraven (lösnings-/systemkrav och komponentkrav/funktionskrav) ska erhållas, med signatur (signoff), innan implementeringsfasen påbörjas. Att uppnå kravöverenskommelse är typiskt den slutliga aktiviteten i projektfasen som innefattar kravanalys och specificering.

Kravöverenskommelsen inbegriper ofta projektets nyckelintressenter, till exempel:

- Projektledare
- Beställare
- Representanter för förvaltning
- Verksamhetsrepresentant från kund
- Affärs- och systemanalytiker
- Kravhanterare
- Representanter för kvalitetssäkring, test och utvecklingsteam

Ett syfte med kravöverenskommelse är att säkerställa att kraven är stabila och att eventuella ändringar hanteras genom formella ändringsbegäranden. Formella överenskommelser minskar risken att nya krav införs under eller efter implementeringen.

Formella överenskommelse kan också minimera risken för missförstånd mellan kunden och leverantören rörande projektets omfattning eftersom alla krav eller krav-/lösningsspecifikationer ska ha granskats noggrant innan signoff.

En kravöverenskommelse kan anses vara komplett när alla relevanta projektintressenter har godkänt kravdokumentet(-n).

När kraven är överenskomna ska detta kommuniceras till projektteamet och är ofta en milstolpe för projektet.

5.3.1 Lösningsmodellering

Lösningsmodellering är ofta en del av kravanalys och specificering då den syftar till att utveckla modeller av verkliga problem som ska lösas av den faktiska affärslösningen. Lösningsmodellering kan också utföras under kravinsamlingen då vissa insamlingstekniker använder olika modelleringsnotationer eller angreppsätt (t.ex. kravinsamling med UML-användningsfall). Den stora utmaningen rörande lösningsmodellering är att välja och skapa lämpliga diagram vilka kan utgöra den modell som beskriver affärslösningen från olika perspektiv. Dessa modeller ska inte bara beskriva affärslösningen korrekt utan också vara lättförståeliga för både kundens och leverantörens representanter.

Lösningsmodellering kan använda många typer av modeller, men generellt sett finns tre grundtyper av modeller:

© SQEB – Software Quality Engineering Board

- Kravmodell – som beskriver behovsområdet och vanligen designas i ett tidigt skede av projektet. Denna modelltyp används främst för kravanalys och estimering av arbetsinsats och tillhandahåller en grund för lösningsmodellen.
- Lösningsmodell – beskriver lösningsområdet från olika perspektiv och fastställer formen för implementering av funktionella och icke-funktionella krav. Affärslösningsmodellen tillhandahåller en grund för lösningsdesignen.
- Konceptuell modell – även känd som domänmodell. En konceptuell modell visar begrepp (entiteter) och relationer mellan dem. Målet med konceptuell modellering är att tydliggöra och förklara innebörden av termer och begrepp som används av domänexperter för att hantera affärsbehovet och fastställa rätt relationer mellan de olika begreppen.

Olika modeller kan användas för ovanstående beroende på vilket perspektiv man vill presentera med den specifika modellen. Följande vyer är vanliga och applicerbara på modellering av behovs- eller lösningsområdet:

- Användarvy (t.ex. modellerad från användningsfall)
- Logisk vy (t.ex. modellerad från de funktionella kraven)
- Processvy (t.ex. modellerad från kommunikation, interaktionsmodeller eller icke-funktionella krav som specificerar effektiviteten hos affärsprocessen)
- Implementationsvy (vanligtvis modellerad från lösningens komponenter)
- Installationvy (t.ex. modellerad från integrationsmodeller och lösningsarkitektur)

Olika nivåer av modellering och olika vyer (perspektiv) på lösningen kan beskrivas genom olika diagram. För att få en komplett bild av lösningen används oftast en kombination av olika vyer. Detta resulterar i att olika modeller/diagram används för att beskriva lösningen från en specifik vy. Några exempel på analysmodeller/-diagram presenteras i tabellen nedan (Tabell 4 Exempel på analysmetoder och -modeller/diagram)

© SQEB – Software Quality Engineering Board

Analysmetod	**Analysmodell/-diagram**
Kontextanalys	Kontextmodell Driftsättningsdiagram (UML)
Logisk analys	Funktionell nedbrytning Kravdiagram (SysML)
Beslutsanalys	Tillståndsmaskindiagram (UML)
Dataanalys	Enhetssambandsmodell (ERM) Klassdiagram (UML)
Användningsfallsanalys	Användningsfallsdiagram (UML)
Processflödesanalys	Affärsprocessmodell (BPMN) Aktivitetsdiagram (UML) Kommunikationsdiagram (UML)

Tabell 3 Exempel på analysmetoder och -modeller/diagram

För kurshållare: förklara fördelar och tillämpning med grundläggande analysmodeller/-diagram (kontextmodeller, funktionell nedbrytning och affärsprocessmodeller) och ge några exempel på användning av dessa diagram.

En vanlig metod för lösningsmodellering av programvara är UML (Unified Modeling Language). UML tillhandahåller många olika typer av diagram för att beskriva olika perspektiv av lösningen. UML är en enhetlig notation för analys och design av system. UML tillhandahåller olika diagram indelade i beteendemässiga och strukturella diagram, där beteendemässiga diagram skildrar beteendemässiga funktioner i ett system eller affärsprocess, och strukturella diagram skildrar de strukturella element som utgör ett system eller en funktion.

© SQEB – Software Quality Engineering Board

Diagramtyp (UML 2.4.1)	Tillämpning
Beteendemässiga diagram	
Aktivitetsdiagram	Modellerar ett systems aktiviteter och det sätt dessa aktiviteter är relaterade i ett övergripande systemflöde. Används vanligen för att modellera procedurer, processflöden och arbetsflöden.
Användningsfallsdiagram	Beskriver användningsfall och relationerna mellan aktörer och systemet. Beskriver de funktionella kraven i ett system, externa aktörers samverkan vid systemgränser och systemsvar.
Tillståndsmaskindiagram	Visar hur ett element kan flytta mellan olika tillstånd och klassificerar dess beteende i enlighet med tillståndsövergångar och begränsningskontroller.
Timingdiagram	Definierar olika objekts beteende inom en viss tidsram. Tillhandahåller en visuell representation över hur objekt ändrar tillstånd och interagerar över tid.
Sekvensdiagram	En strukturerad bild av beteenden som en serie av sekventiella steg över tid. Används för att illustrera arbetsflöden, meddelandeflöden (kommunikation) och hur element samarbetar generellt över tid för att uppnå ett resultat.
Kommunikationsdiagram	Visar samarbete mellan element under körning (i run-time) för att visa samband mellan olika objekt.
Interaktionsöversikts-diagram	Visualiserar samarbetet mellan olika interaktionsdiagram för att illustrera ett övergripande kontrollflöde.

© SQEB – Software Quality Engineering Board

Diagramtyp (UML 2.4.1)	Tillämpning
Strukturella diagram	
Klassdiagram	Fångar den logiska strukturen hos systemet, klasserna och objekten som skapar den modell som beskriver vad som finns och vilka attribut och beteenden systemet har. Kan användas för att modellera data.
Objektdiagram	Visar instanser av klasser och relationer mellan dessa vid en given tidpunkt.
Paketdiagram	Skildrar hur modellelement är organiserade i paket och beroenden mellan dessa. Kan användas för att organisera krav utifrån typ/nivå av abstraktion.
Komponentdiagram	Visar de programvarudelar som utgör ett system samt hur de är organiserade och beroenden mellan dem.
Driftsättningsdiagram	Visar systemets driftsarkitektur.
Sammansättnings-/strukturdiagram	Visar det interna samarbetet mellan klasser, gränssnitt och komponenter (med deras egenskaper) för att beskriva en viss funktionalitet.
Profildiagram	Gör det möjligt att skräddarsy UML-metamodellen för olika plattformar (som J2EE eller .NET) eller domäner (som realtid- eller verksamhetsprocess-modellering).

Tabell 4 UML-diagram och deras tillämpning

För att modellera mer komplexa lösningar, särskilt vid systemutveckling, kan en annan modelleringsnotation användas – SysML (System Modeling Language). SysML tillåter modellering av en mängd olika system där hårdvara, mjukvara, information, processer, personal och hjälpmedel ingår. SysML återanvänder sju av UMLs diagram och tillhandahåller två nya diagram: ett kravdiagram vilket innehåller krav på funktioner, prestanda och gränssnitt samt ett parametriskt diagram för att definiera prestanda och kvantitativa begränsningar.

För att modellera flöden i affärsprocesser och kommunikation mellan verksamhetsaktörer kan andra modelleringsnotationer användas – till exempel BPMN (Business Process Modeling Notation) vilket ofta används i tidiga steg av kravanalysen eller under analys av affärsprocesser som utförs i kunden organisation.

För att modellera aspekter runt användargränssnitt hos den planerade lösningen (särskilt när det gäller programvarulösningar), kan GUI-modellering användas. Det görs vanligen genom så kallade GUI-prototyper och tillhandahåller ett sätt att designa och testa delar i ett användargränssnitt. GUI-prototyper är särskilt användbart i situationer när kraven inte är tydliga och kan implementeras på

© SQEB – Software Quality Engineering Board

många olika sätt, eftersom de tillåter intressenterna att välja det lämpligaste alternativet för implementation.

Följande varianter av prototyper kan särskiljas [IBUQ]:

- Vertikala prototyper: reducering till ett fåtal individuella, men detaljerade funktioner
- Horisontella prototyper: så många funktioner som möjligt integrerade med varandra, men inte funktionella (används mest för gränssnittstestning)
- Scenario-prototyper: alla funktioner simuleras för en specifik aktivitet genom en kombination av vertikala och horisontella prototyper

Beroende på syftet med användandet skapas prototyper av olika typer och varianter [IBUQ]:

- Low fidelity (lo-fi) -prototyper – liten likhet med slutprodukten; utvärdering av fördelarna med idén
- High fidelity (hi-fi) -prototyper – stor likhet; utvärdering av detaljer och exakta funktioner
- Hybridformer – low-high fidelity (lo-hi-fi) -prototyper, till exempel interaktiva simuleringar genom HTML eller PowerPoint®

För kurshållare: förklara fördelar och tillämpning av grundläggande UML-diagram (aktivitetsdiagram, användningsfallsdiagram, tillståndsmaskindiagram och klassdiagram) och ge exempel på hur dessa diagram kan användas)

© SQEB – Software Quality Engineering Board

5.4 Kravspecificering	60 minuter

Kravutveckling är inte enbart en process att identifiera och analysera krav, det är också en process att främja en effektiv kommunikation mellan olika intressenter. Det sätt på vilket kraven dokumenteras spelar en viktig roll i att säkerställa att de kan läsas, ändras och valideras. Att dokumentera krav kallas kravspecificering.

En specifikation definieras som en explicit uppsättning krav som ska tillfredsställas av ett material, en produkt eller tjänst; därför kan det ses som en sorts kontrakt som beskriver lösningens beteende och funktioner. Specifikationen används för att spåra och hantera krav. I specifikationen specificeras krav på ett strukturerat sätt och modelleras separat (krav modelleras ”oberoende” av varandra, när högnivå-krav bryts ner till en nivå där varje individuellt krav bildar en ”oberoende” enhet som ytterligare kan utvecklas och spåras). Specifikationen dokumenteras som en formell överenskommelse avseende vilka krav som ska implementeras i det planerade systemet (eller i andra former av lösningen); därför kan kravspecifikationen ses som en del av slutresultatet av kravanalysen.

Termen “specifikation” kan användas i både krav- och lösningssammanhang.

Kravspecificering beskriver det aktuella behovsområdet (ett förslag på affärslösning för ett givet affärsproblem, behov eller mål, en ny funktionalitet, etc.) och innehåller minst följande information:

- Affärskrav tillsammans med dess acceptanskriterier
- Begränsningar och antaganden

Att ta fram kundens kravspecifikation bör vara kundens uppgift. I vissa fall kan dock leverantören hjälpa till med detta arbete.

En kravspecifikation måste inte vara ett formellt ”specifikationsdokument”. Den kan till exempel vara en sprintbacklog eller en uppsättning krav som underhålls i ett kravhanteringsverktyg.

Framtagande av andra typer av specifikationer för krav såsom systemkrav, programvarukrav, säkerhetskrav, säkerhetskritiska krav, miljökrav, lagkrav, etc. är uppgifter för andra roller.

Lösningsspecifikationen beskriver lösningen från olika perspektiv. Specifika typer av lösningsspecifikationer är: funktionell kravspecifikation, systemkravspecifikation eller programvaruspecifikation.

En funktionell specifikation kan vara ett dokument som tydligt beskriver de tekniska krav som ska lösas. Den funktionella specifikationen är grunden för vidare produktutveckling och testning. Den måste tillhandahålla precis information rörande alla funktionella aspekter på den programvara som ska implementeras. En funktionell specifikation beskriver inte hur de önskade funktionerna hos programvaruprodukten ska implementeras och den beskriver inte vilken teknik som ska användas. Den fokuserar istället på funktionalitet och interaktioner mellan användaren och produkten.

© SQEB – Software Quality Engineering Board

Både krav- och lösningsspecifikationer kan skrivas i form av användningsfall (verksamhetsanvändningsfall eller systemanvändningsfall, beroende på specifikationens syfte och målgrupp).

En typ av specifikation kan vara en användarberättelse. Användarberättelser används ofta i agila utvecklingsmetoder. Användarberättelser är ett snabbt sätt att hantera kund- eller användarkrav. Avsikten med en användarberättelse är att snabbt och med mindre overheadkostnader kunna hantera föränderliga verklighetsförankrade krav.

En användarberättelse beskriver den funktionalitet som kommer att ge kundnytta. Den har tre aspekter [Cohn]:

- En skriftlig beskrivning av användarberättelsen som används för planering och som en påminnelse (oftast i form av en förklaring ”Som en [slutanvändare], vill jag [önskemålet] så att [motivet]”)
- Konversation runt användarberättelsen för att komplettera med detaljer
- Test som förmedlar och dokumenterar detaljer och används för att avgöra när en användarberättelse är komplett

Användarberättelser används ofta tillsammans med personas (d.v.s. arketypiska egenskaper) som representerar en specifik typ av slutanvändare.

Arbetet med specificering kan stödjas av standarder som ger riktlinjer rörande hur specifika dokument kan skrivas, typiskt innehåll och allmän praxis. Standard som kan användas vid framtagande av specifikationer är:

- ISO/IEC/IEEE 29148. Systems and software engineering -- Life cycle processes -- Requirements engineering (Ersätter IEEE 830, IEEE 1233 och IEEE 1362)

I allmänhet fungerar en specificering som en aktivitet för att formalisera kravanalysens resultat. Identifiering, analys och specificering leder fram till kravöverenskommelse (se 3.3.2 Kravanalys).

Proceduren för att skapa en lösningsspecifikation består vanligen av följande aktiviteter:

1. Definition av vision och mål
    - Vad är målet med lösningen eller det specifika lösningsområdet som dokumenteras i lösningsspecifikationen?
2. Identifiering av intressenter
    - Vem kommer att använda produkten?
    - Vem är ansvarig för det specifika lösningsområdena och kan stödja eller påverka skapandet av lösningsspecifikationen?
3. Fastställande av krav
    - Vilka högnivå-krav (på affärsnivå) ska täckas av specifikationen?
    - Hur är kraven prioriterade?

© SQEB – Software Quality Engineering Board

4. Strukturerad kravspecifikation
    - Vilka beroenden finns mellan krav?
    - Hur kan krav organiseras?
5. Beskrivning av systemmiljön
    - I vilket sammanhang ska lösningen fungera?
6. Fastställande av lösningen
    - Definition av lösning och omfattning med relevanta aspekter som inte ingår i, men som kan påverka omfattningen, såsom gränssnitt mot externa system
7. Kravanalys
    - Detaljerad kravanalys (nedbrytning av affärskrav till lösnings- eller systemkrav men även funktions- eller komponentkrav på lägre nivå)
    - Identifiering och lösande av potentiella inkonsistensproblem mellan krav
    - Mappning av affärskrav mot lösnings- och systemkrav och skapa en struktur för spårbarhet
8. Modellering av affärsbehoven
    - Modellering av de affärsbehov som ska uppfyllas med hjälp av lösningen (inklusive modellering eller förbättring av existerande affärsprocesser i förekommande fall). Ett modelleringsspråk för affärsprocesser används ofta för att beskriva affärsmodellen (t.ex. BPMN)
    - Hur lösningen kommer att infogas i den existerande infrastrukturen
9. Modellering av lösningen
    - Modellering av lösningen. Kan inkludera egenskaper hos programvaran, hårdvaran, tjänster, processer, etc. beroende på typ av lösningsspecifikation
    - Modelleringsnotation för att visa lösningen från olika perspektiv (t.ex. UML)

Ovanstående procedur involverar ett antal intressenter som stöder specificeringsarbetet inom olika områden och formaliserar resultatet av kravanalysprocessen. Individuella steg inom proceduren kan utföras parallellt, iterativt eller även hoppas över beroende på vilken information som redan samlats in och valt angreppssätt för kravhantering. I t.ex. agila metoder är proceduren för lösningsspecificering inte lika formell eftersom krav och lösning vanligtvis beskrivs på ett mer lättviktigt sätt. Resultatet från proceduren, lösningsspecifikationen, fungerar som startpunkt för mjukvaru-, hårdvaru- och databasdesign. Den beskriver systemets funktionalitet (funktionella och icke-funktionella specifikationer), systemets prestanda och begränsningar rörande drift och användargränssnitt.

Produktdokumentation, inklusive krav- eller lösningsspecifikationer, kan skapas med olika formaliseringsgrad:

© SQEB – Software Quality Engineering Board

- Icke-formell
- Halvformell
- Formell

Ett icke-formellt angreppssätt att skriva en specifikation betyder att dokumentet är skrivet med vardagligt språk, utan någon formell notation. Detta angreppssätt kan användas när läsarna inte har någon erfarenhet av mer formella eller tekniska specificeringsspråk och därmed kan ha svårt att förstå dokumentets innehåll. Den största svagheten med detta angreppssätt är att det är otydligt och kan leda till missförstånd och feltolkningar. Dessutom är icke-formella specifikationer inte en bra grund för implementering och testning eftersom de inte är tillräckligt tydliga och precisa.

En halvformell specifikation innehåller viss formell notation och blir mer strukturerad. Vanligtvis baseras sådana specifikationer på specifika mallar (ofta hämtade från relevanta standarder). Halvformella specifikation kan uttrycka krav i form av modeller och använda ett mer formaliserat vardagligt språk. En av de vanligaste notationerna som användes för halvformell dokumentation är UML.

En formell specifikation är en matematiskt och/eller algoritmisk beskrivning av en lösning som kan användas för att utveckla en implementation. En formell specifikation beskriver vad produkten ska göra och vanligtvis inte hur det ska utföras. Eftersom formell specificering baseras på matematiska formler eller ett specifikt språk (t.ex., VDM Specification Language (VDM-SL)) är det svårare att lära sig, ganska sällsynt och kräver specifika kunskaper.

© SQEB – Software Quality Engineering Board

5.5 Validering och verifiering av krav	40 minuter

Eftersom krav är grunden för systemutveckling, kommer varje felaktigt eller saknat krav att fortplanta sig till andra utvecklingsprocesser i projektet. Det är viktigt att notera att fel som beror på dålig kvalitet i kraven generellt är dyrare att åtgärda i senare faser av projektet än andra typer av fel. Dessutom, ju senare fel upptäcks, desto dyrare är det att åtgärda dem.

Därför är nödvändigt att verifiera (“producerar vi produkten rätt”) och validera (”producerar vi rätt produkt”) kraven. Validering och verifiering av krav ska ske kontinuerligt under utvecklingen av lösningen för att säkerställa att den produkt som utvecklas möter ställda kvalitetskriterier och kommer att uppfylla intressenternas behov. Bästa sättet är att planera och utföra validering och verifiering av krav redan i de tidiga faserna av lösningsutvecklingen – i idealfallet vid kravinsamlingen. Vanliga tekniker för validering och verifiering inkluderar olika typer av granskningar, prototyparbete och presentationer av den föreslagna lösningen, eller kraven, för intressenter med målet att få återkoppling.

Validerings- och verifieringsaktiviteter bör också säkerställa att kraven och/eller krav-lösningsspecifikationer överensstämmer med företagsstandarder (mallar) och är dokumenterade och sedan testade mot kvalitetskriterierna (se 1.1.4 Kravkvalitet). Det är också viktigt att validera de modeller som utvecklas under kravanalys- och specificeringsaktiviteterna.

Eftersom krav är grunden för utveckling och testning av lösningen är deras kvalitet av yttersta vikt för att projektet ska bli framgångsrikt. Tydliga, kompletta, konsistenta och testbara krav minskar risken för att produkten är misslyckad (eller ett misslyckat projekt), eftersom de möjliggör noggrann testning. Därför rekommenderas att involvera testare i granskning av kraven eftersom de avsevärt kan bidra till att förbättra kvaliteten hos kraven och/eller lösningsspecifikationer genom att identifiera svagheter och möjliga fel.

Testbarheten hos krav stöds av acceptanskriterier. Acceptanskriterier beskriver vad som måste vara uppfyllt för att lösningen ska kunna godkännas och dessa ska vara överenskomna från bägge sidor, kunden och leverantören, innan projektstarten. Det bästa sättet att definiera acceptanskriterier är att de ingår som en del i kontraktsdokumentation. Varje högnivåkrav måste ha minst ett acceptanskriterium och varje kriterium måste vara mätbart på ett realistiskt och överenskommet sätt. Sådana kriterier utgör ofta grunden för projektets kvalitetsplan och acceptanstestning.

För kurshållare: ge exempel på acceptanskriterier för exempelkrav.

© SQEB – Software Quality Engineering Board

6 Kravhantering i modeller	100 minuter

Begrepp

Agilt angreppssätt, CMMI, Iterativt angreppssätt, Mognadsmodell, Processmodell, Produktbacklog, Produktlivscykel, Produktägare, Rational Unified Process, Sekventiellt angreppssätt, SPICE

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska inneha efter att slutförandet av respektive avsnitt.

6.1 Utvecklings- och förvaltningsmodeller och angreppssätt

LO-6.1.1 Förstå de vanligaste egenskaperna hos en produktlivscykelmodell (K2)

LO-6.1.2 Förstå likheter och skillnader i kravhanteringsprocessen mellan modeller som används inom produktutveckling och förvaltning (K2)

6.2 Mognadsmodeller

LO-6.2.1 Förstå hur kravhantering är en del av vanliga mognadsmodeller (K2)

© SQEB – Software Quality Engineering Board

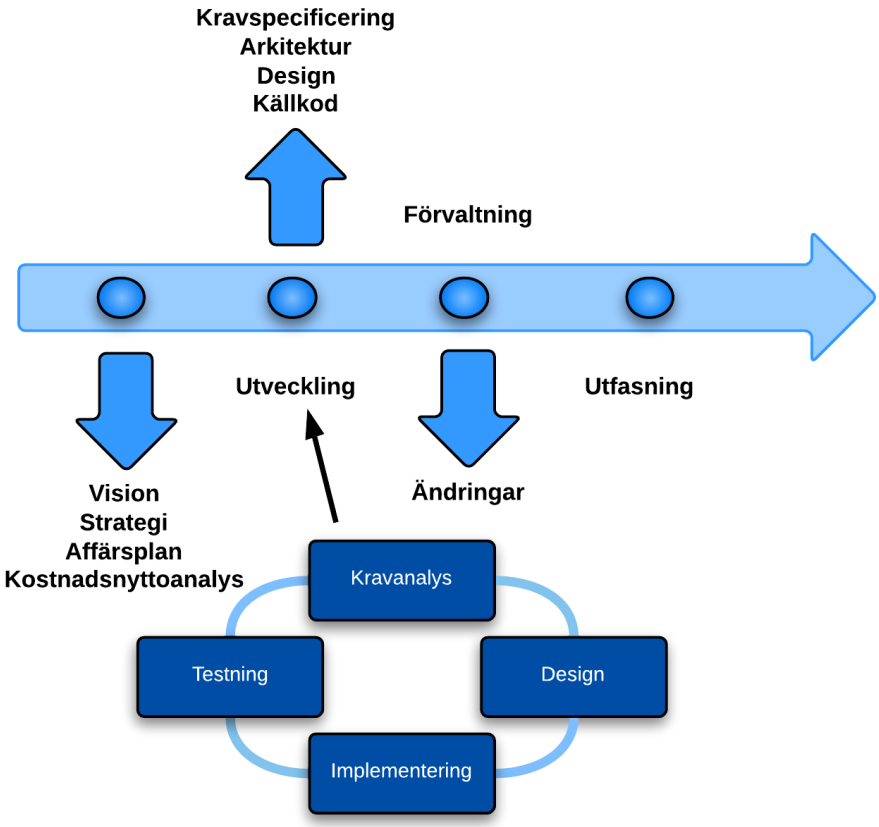
6.1 Utvecklings- och förvaltningsmodeller och angreppssätt	60 minuter

Strukturen hos kravhanteringsprocessen beror ofta på vilken utvecklingsmodell som används. Den generiska kravhanteringsprocessen ska anpassas till de specifika behoven och utvecklingsmiljön.

Syftet med detta kapitel är att tillhandahålla exempel på hur olika kravhanteringsaktiviteter går till i några olika typer av utvecklingsmodeller.

Processmodeller är metodoberoende och beskriver utvecklingsprocesser. Processmodeller beskriver vanligen roller, aktiviteter, faser och dokument och ger en ger ett standardformat för planering, organisering och genomförande av ett projekt.

En generell produktlivscykel (PLC) beskriver olika faser av produktutveckling där de grundläggande faserna är planering, utveckling, förvaltning och utfasning. Utvecklingsfasen delas ofta in i flera faser som ofta utförs iterativt (se figur 2 Grundläggande produktlivscykel (PLC)).



Figur 2 Grundläggande produktlivscykel (PLC)

© SQEB – Software Quality Engineering Board

I allmänhet kan utvecklingsmodeller klassificeras som traditionella angreppssätt eller agila angreppssätt. Traditionella angreppssätt kan delas in i sekventiella modeller, iterativa modeller och inkrementella modeller (de två sistnämnda kombineras ofta med varandra).

Den enklaste utvecklingsmodellen, vattenfallsmodellen, kan ses som en linjär representation av den generiska kravhanteringsprocessen, baserat på antagandet att det är möjligt att samla in, definiera och analysera alla krav i ett tidigt skede av utvecklingen och att de inte kommer att ändras. Den stora svagheten med denna processmodell är svårigheten med anpassningar till ändringar. Denna svaghet har gjort att den slutat användas i många moderna projekt där krav ändras med anledning av förändringar i verksamhet, teknik eller marknad.

Den generella V-modellen är en vidareutveckling av den ursprungliga vattenfallsmodellen. Den huvudsakliga förbättringen som implementerats genom V-modellen är tydliggörandet mellan utvecklingssteg och teststeg, där varje utvecklingssteg har ett motsvarande teststeg. Modellen kan ses som en nedbrytning av krav där varje nivå har en tillhörande testnivå.

Denna modell är tämligen effektiv för kravhantering i utvecklingsfasen och ger en bra grund för kravledning med spårbarhet och ändringshantering. Kravutveckling i V-modellen kan dock vara svårt att genomföra eftersom det ofta är omöjligt att säkerställa att alla viktiga krav identifieras i tidiga skeden av projektet. Omfattningen av produkten blir mycket ofta större på grund av krav som kommer från olika intressenter och omgivande system eller miljöer i senare faser. Detta leder till behov av omfattande omarbete och ändringhantering.

V-modellen kan anses vara en av de bästa modeller för både kravutveckling och kravledning för väldefinierade och stabila områden.

För kurshållare: grafisk återgivning av den generella V-modellen; beskrivning av kravhanteringsprocessen i den generella V-modellen

I de fall där det är fråga om mer komplexa eller instabila lösningar rekommenderas att använda iterativa eller inkrementella modeller. Det iterativa/inkrementella angreppssättet bygger på det faktum att krav ofta är svåra att specificera, särskilt i kommersiella system. Tekniker för iterativ och inkrementell utveckling förutsätter samarbete och partnerskap mellan utvecklare och användare. Krav definieras ofta enbart i termer av affärsmål – dessa affärsmål vidareutvecklas sedan i senare iterationer.

Vid inkrementell utveckling görs antagandet att kravlistan delas upp i mindre delar och varje del utvecklas i steg eller inkrement. Tanken är att varje del ska kunna implementeras och, i bästa fall, driftsättas så att ledtiden till marknaden hålls kort. Det är också möjligt att få direkt feedback på de inkrement som levererats, så att kvarvarande, ännu ej utvecklade inkrement, kan förbättras. På detta sätt kan risker reduceras (särskilt jämfört med en ”allt på en gång”- stategi). Även om projektet avslutas innan det är färdigställt kan intressenter få ut vissa fördelar de redan kompletta inkrementen.

Det iterativa angreppssättet är särskilt användbart för stora projekt med ett stort antal krav. Huvudtanken med det iterativa angreppssättet är att utveckla en lösning genom upprepade cykler (som kan ses som mini-PLC) och i mindre delar i taget. Syftet med att organisera arbetet på det sättet är att göra projektet lättare att kontrollera och för att kunna övervaka och hantera risker på

© SQEB – Software Quality Engineering Board

ett systematiskt sätt. Ett exempel på en iterativ modell är RUP©, Rational Unified Process, en processmodell från IBM Rational.

RUP är en modell för programvaruutveckling i ett objekt-orienterat sammanhang. Projektet organiseras i fyra huvudfaser, förberedelse, etablering, konstruktion och överlämning. Var och en av dessa faser har sitt eget syfte och produkter, som definieras av RUP:s ramverk. RUP är en iterativ modell där innehållet i varje iteration väljs ut baseras på risken hos den uppgift som ska implementeras. Antalet iterationer i en specifik fas kan skilja sig åt och justeras utgående från projektets eller produkten specifika behov. Kravhanteringsarbetet täcks i allmänhet av krav- och analys och design-disciplinerna som definieras i ramverket för RUP.

För kurshållare: fördjupning i RUP© med en grafisk presentation; fördjupad studie av kravdisciplinen.

Alla processmodeller som beskrivs ovan tillhör så kallade traditionella angreppssätt. Den vanligaste utmaningarna för traditionella angreppssätt är att stödja frekventa ändringar. De flesta traditionella modeller stödjer inte sådana ändringar på ett effektivt sätt. Som en lösning på detta problem uppstod så kallade agila modeller. Agila angreppssätt baseras på det agila manifestet vilket innehåller en uppsättning värderingar och principer som används i agila miljöer. De viktigaste principerna i det agila manifestet är [Agile Manifesto]:

- Kundnöjdhet genom snabb leverans av användbar programvara
- Välkomna förändrade krav, även sent under utvecklingen
- Leverera fungerande programvara ofta (veckor snarare än månader)
- Enkelhet – konsten att maximera mängden arbete som inte görs – är grundläggande
- Regelbunden anpassning till ändrade omständigheter

Dessa principer ger ganska betydande konsekvenser på kravhanteringen. I agila miljöer kommuniceras och spåras krav ofta genom mekanismer som produktbackloggar, användarberättelser (user stories), story cards eller skärmprototyper. Kravåtagandet görs antingen kollektivt av teamet eller av en teamledare med befogenhet att göra det. Spårbarhet och konsistens mellan krav och arbetsprodukten kan hanteras genom de mekanismer som redan nämnts (produktbackloggar, användarberättelser, story cards och skärmprototyper) såväl som genom aktiviteter i början och slutet av iterationen såsom ”retrospektiv” och ”demodagar”. Kravledningsprocessen är därför inte lika formell som den är i traditionella utvecklings- eller förvaltningsmodeller.

Processen för kravutveckling är tät förknippad med den specifika beskrivningen och analys av användarberättelser eller scenarier. Många aktiviteter utförs normalt under kravutvecklingen och beskrivs inte i agila modeller men förväntas utföras av det agila utvecklingsteamet under produktägarens ansvar.

I agila miljöer blir kundens behov och idéer iterativt insamlade, detaljerade, analyserade och validerade. Krav kan samlas i backloggar och specificeras i form av användarberättelser, scenarier eller användningsfall. Vilka krav som kommer att hanteras i varje given iteration styrs av en bedömning av risk och av de prioriteringar som gjorts för det som kvarstår i produktbackloggen. På

© SQEB – Software Quality Engineering Board

vilken detaljnivå kraven (och andra artefakter) ska dokumenteras styrs av behovet av koordination (mellan teammedlemmar, team och senare iterationer) och risken att förlora den inhämtade kunskapen.

Exempel på agila angreppssätt är Extreme Programming och Scrum.

Extreme Programming (XP) är en metod för utveckling av programvara som tagits fram av Kent Beck m.fl. med målet att bland annat kunna hantera förändringar i kundkraven. XP förespråkar täta leveranser av programvara till kunden i korta utvecklingscykler (time boxing), vilket syftar till att öka produktiviteten och sätta upp kontrollpunkter där nya krav kan tas in.

Några kännetecken för Extreme Programming:

- Undvik att programmera features tills de faktiskt behövs
- Räkna med ändringar i kundens krav vartefter tiden går och behovet blir tydligare
- Kontinuerlig kommunikation med kunden och mellan programmerare
- Fullständigt avstånd till fastställande av krav (ingen separat krav-”fas” innan utvecklingen startar; kravutveckling, detaljering och upptäckande är en del av den faktiska programvaruutvecklingen (programmeringen))

Scrum är ett agilt ramverk som består av uppsättningar av arbetssätt och fördefinierade roller. En av huvudegenskaperna i Scrum är att utvecklingen delas upp i ”sprintar” (vanligen mellan två och fyra veckor långa). I varje sprint skapar teamet ett så kallat ”potentiellt leveransklart produktinkrement”.

Scrumprojekt hanterar krav via “backloggar”. Det finns två typer av backloggar:

- Produktbacklog – en högnivålista som underhålls under hela projektet. Den summerar krav i form av övergripande beskrivningar av alla potentiella features, prioriterade efter affärsnytta. Produktbackloggen är produktägarens egendom.
- Sprintbacklog – listan som ska behandlas av teamet under nästa sprint. Features bryts ner till aktiviteter vilka normalt bör omfatta vara mellan fyra och sexton timmars arbete. Sprintbackloggen är teamets egendom.

Scrums huvudsakliga påverkan på kravhantering betyder att kravspecifikationer inte är kompletta och validerade innan utvecklingen påbörjas.

Produktägaren och teamet kommer överens om vilka features från produktbackloggen som ska inkluderas i kommande sprint baserat på affärsprioritet och den arbetsinsats som krävs. Användarkrav formuleras av produktägaren som ”användarberättelser” (user stories) vilka innehåller information om ett kravs ”vem, vad och varför” men inte ”hur”. I sprintens början bryts de utvalda kraven (features) ned till aktiviteter i sprintbackloggen och utvecklas sedan.

Produktägarnas medverkan, till exempel genom att titta på presentationer av funktioner som implementerats i programvaran, hjälper till att tydliggöra kraven för teamet och för produktägarna själva.

© SQEB – Software Quality Engineering Board

För kurshållare: Ge exempel på användarberättelser (user stories) och artefakter i produkt- och sprintbacklog.

© SQEB – Software Quality Engineering Board

6.2 Mognadsmodeller	40 minuter

Kravhanteringsprocessen kan förbättras precis som vilken alla andra processer som definieras som en del av utveckling eller förvaltning av en produkt eller lösning. Kravinsamling kan förbättras genom insamling av kundkrav på ett mer effektivt sätt. För att kunna åstadkomma detta kan organisationen införa tekniker som säkerställer att kraven samlas in snabbare, är kompletta och kan komma överens om med de flesta intressenter. Dessa tekniker inkluderar intervjuer, brainstorming, initialt arbete med prototyper, användande av personas, samt scenarier.

En vanlig metod med förbättringar av kravhanteringsprocessen är att använda mognadsmodeller. Dessa modeller kräver bedömning av nuvarande nivå att processmognad och identifiera områden som kan förbättras. Mognadsmodeller använder ofta mognadsnivåer. Mognadsnivåer används för att identifiera, bedöma och förbättra processmognaden.

Exempel på mognadsmodeller som kan användas för kravhanteringsprocessen är ISO/IEC 15504 (SPICE - Software Process Improvement and Capability Determination) eller Capability Maturity Model Integration (CMMI). Båda modeller definierar fem mognadsnivåer för specifika processer eller områden och möjliggör (tillåter) jämförelse av mognad mellan olika organisationer.

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination), är en uppsättning tekniska standarder för programvaruutvecklingsprocessen och relaterade verksamhetsledningsfunktioner. SPICE kan användas som ett verktyg för processförbättring och/eller fastställande av kapacitet (till exempel, utvärdering av leverantörens processkapacitet). SPICE definierar fem mognadsnivåer för följande processer: Kund-Leverantör, Utveckling, Stödjande, Ledning och Organisation. Dessa fem nivåer är:

1 Utförd process (Performed process)

2 Hanterad process (Managed process)

3 Etablerad process (Established process)

4 Förutsägbar process (Predictable process)

5 Optimerad process (Optimizing process)

De viktigaste processområdena relaterade till kravhantering i SPICE-modellen är:

- Förberedelse
- Kravfångst
- Analys av systemkrav
- Analys av programvarukrav

© SQEB – Software Quality Engineering Board

Capability Maturity Model Integration (CMMI) definierar fem mognadsnivåer för utveckling, tjänster och anskaffningar:

1. Initial (Initial)
2. Hanterad (Managed)
3. Definierad (Defined)
4. Kvantitativt hanterad (Quantitatively managed)
5. Optimerad (Optimizing)

De viktigaste processområdena kopplat till kravhantering i CMMI är:

- Kravledning (REQM Requirements Management)
- Kravutveckling (RD Requirements Development)

Kravhanteringsprocessen i sig kan ses som ett medel för förbättring av utvecklingsprocesser eftersom effektiv kravhantering förbättrar implementeringsinsatsen genom att eliminera att onödig tid läggs på att tydliggöra och förklara krav. Dessutom reduceras antalet defekter som beror på fel i kravspecifikationer och/eller missförstånd rörande kravdokumentationen på grund av dålig kvalitet. Bra kravhanteringsprocesser ökar kvaliteten på testarbetet genom att tillhandahålla väl beskrivna, precisa och testbara krav och gör acceptanstestningen enklare och med tillförlitlig eftersom basen för sådan testning vanligen är kraven.

För kurshållare: Visa de typiska kraven för kravhanteringsprocessen i processmognadsmodeller

© SQEB – Software Quality Engineering Board

7 Verktygsstöd	80 minuter

Begrepp

Avvikelsehanteringsverktyg, Kravinsamlingsverktyg, Kravhanteringsverktyg, Modelleringsverktyg, Pilotprojekt, Projektstyrningsverktyg, “Proof of concept”, Prototypverktyg, Verktygskategorier, Ändringshanteringsverktyg

Inlärningsmål för Kravhantering, grundnivå

Målen fastställer vilka kunskaper du ska inneha efter att slutförandet av respektive avsnitt.

7.1 Fördelar med verktyg

LO-7.1.1 Förklara fördelarna med att använda verktyg för att stödja olika kravhanteringsaktiviteter (K2)

7.2 Verktygskategorier

LO-7.2.1 Förstå syfte och användande av olika kategorier av verktyg som stödjer kravhantering (K2)

7.3 Välja verktyg

LO-7.3.1 Förklara viktiga faktorer som ska betraktas när verktyg för kravhantering väljs (K2)

LO-7.3.2 Förklara processen att införa ett verktyg i en organisation (K2)

© SQEB – Software Quality Engineering Board

7.1 Fördelar med verktyg	30 minuter

Det finns olika verktyg som stödjer specifika kravhanteringsaktiviteter.

Ett vanligt användningsområde för verktyg är att stödja lagring och hantering av krav. Dessa tillhandahåller en gemensam lagringsplats för krav, där krav kan organiseras och spårbarhet underhållas. Det är därmed möjligt att hålla komplexa och föränderliga dokument konsistenta och uppdaterade. Dessa verktyg kan automatisera vissa manuella aktiviteter och även tillhandahålla statistik och rapporter av olika slag. Modelleringsverktyg modellerar krav enligt specifika notationer, såsom UML eller BPMN. Modelleringsverktyg kan ofta generera kravspecifikationerna utifrån modellerna.

Verktyg kan i allmänhet stödja följande aktiviteter inom kravhantering:

- Kravinsamling och lagring
- Definiera och underhålla kravspårbarhet
- Krav- och lösningsmodellering (inklusive prototyper och affärsmodeller)
- Dokumentation av krav (skapande av kravspecifikationer)
- Statusrapportering
- Konfigurations- och ändringhantering

Några fördelar med att använda verktyg är följande:

- Säkerställa att alla krav lagras på ett och samma ställe och är åtkomliga för alla involverade intressenter
- Stödjer kravspårbarhet (t.ex. till testfall), vilket kan användas för att verifiera den relevanta kravtäckningen
- Hantera kravändringar på ett enkelt sätt som inkluderar konfigurationskontroll
- Förbättra kvaliteten på kravspecifikationer genom att använda definierade dokumentmallar och modelleringsnotation
- Spara tid genom att automatisera vissa aktiviteter (såsom generering av kompletta specifikationer från verktyget).
- Tillhandahålla information för beslutsfattande (till exempel, mätetal som indikerar antalet implementerade krav).

© SQEB – Software Quality Engineering Board

7.2 Verktygskategorier	20 minuter

Verktyg som stödjer kravhanteringsaktiviteter kan klassificeras enligt följande:

- Kravhanteringsverktyg
- Kravinsamlingsverktyg
- Kravmodelleringsverktyg
- Prototypverktyg
- Felhanteringsverktyg
- Konfigurations- och ändringshanteringsverktyg
- Projekthanteringsverktyg

Många verktyg stödjer mer än en aktivitet, till exempel kan modelleringsverktyg tillhandahålla en lagringsplats för krav med stöd för konfigurations- och ändringshantering, stödja olika modelleringsnotationer, generering av dokumentation och statistik.

För kurshållare: visa ett exempel på ett verktyg som stödjer minst två kravhanteringsaktiviteter och ge några praktiska exempel på tillämpning.

© SQEB – Software Quality Engineering Board

7.3 Val av verktyg	30 minuter

Val av verktyg måste göras innan produkten utvecklas för att undvika de problem som kan uppstå om man byter verktyg mitt i projektet.

Kostnaden för att köpa ett verktyg varierar stort. Kommersiella verktyg kan vara mycket dyra medan open source-verktyg kan vara gratis. Valet var verktyg måste göras mycket noggrant. Innan ett verktyg väljs bör en analys genomföras. Analysen kan beakta följande frågor:

- Bedömning av organisationens mognad, styrkor och svagheter och identifiering av möjligheter för en förbättras kravhanteringsprocess med stöd av verktyg
- Krav rörande de funktioner som krävs från verktyget
- En ”proof-of-concept” genom användande verktyget under en utvärderingsfas med målet att fastställa om verktyget fungerar effektivt med den givna kravhanteringsprocessen, med existerande infrastruktur och för att identifiera eventuella ändringar som kan behöva göras i nuvarande infrastruktur
- Estimering av en kostnads-nyttokalkyl baserat på ett konkret affärsexempel vilket kan inkludera analys av kostnaden för verktyget för ett specifikt projekt eller för alla/de flesta projekten
- Integration mellan ett kravhanteringsverktyg och andra nödvändiga verktyg (till exempel felhanteringsverktyg, projektstyrningsverktyg)
- Behovet av kommunikation mellan ett kravhanteringsverktyg och ett verktyg som redan används i kundens organisation (i vissa fall genomför kunden sin egen kravanalys och dessa resultat kan sedan migreras till leverantörens miljö)
- Hur lätt verktyget är att använda och lära sig (potentiell kostnad för utbildning), tillgång till hjälp online, manualer, guider, och annat stöd

Introducering av det valda verktyget i en organisation bör börja med ett pilotprojekt. Vanliga mål med ett pilotprojekt kan vara följande:

- Mer detaljerad kunskap om verktyget
- Utvärdera hur verktyget passar in med existerande processer och arbetssätt, och bestämma vad som skulle behöva ändras
- Fastställa standardiserade sätt att använda, hantera, lagra och underhålla verktyget och kravhanteringsprodukterna

Ett pilotprojekt kan sänka riskerna vid införandet av ett verktyg. Görs det forcerat och utan lämplig analys, kan det resultera i höga kostnader. Nedan är några exempel på kostsamma verktygsproblem:

- Val av ett verktyg som inte uppfyller intressenternas krav och inte uppfyller sitt syfte

© SQEB – Software Quality Engineering Board

- Inköp av ett dyrt verktyg som används bara en gång i ett projekt eller inköp av ett dyrt verktyg när det finns open source-verktyg med motsvarande funktionalitet
- Utbildning som krävs för ett verktyg som saknat tillräckliga hjälpsystem (särskilt när enbart grundläggande funktionalitet behövs från verktyget)
- Utökning av det valda verktyget med tilläggsfunktioner som inte stöds av verktyget (när det finns andra verktyg med sådan funktionalitet)
- Integrering av verktyget med andra verktyg som används i organisationen

Generella framgångsfaktorer för användande av verktyget kan vara följande:

- Anpassa och förbättra processer för att bättre passa användande av verktyget
- Stegvis utrullning i organisationen
- Tillhandahåll utbildning och stöttning/mentorskap för nya användare, inklusive att fastställa riktlinjer för användandet
- Implementera ett sätt att samla in information från den faktiska användningen (inklusive mätetal och erfarenheter)
- Övervaka användande och fördelar med verktyget

© SQEB – Software Quality Engineering Board

8 Tabeller/Bilder

Tabeller

Tabell 1 Mål med programmet på grundnivå, dess fördelar och huvudfokus 7
Tabell 2 Översikt till kravhanteringsaktiviteter, input och resultat..... 31
Tabell 4 Exempel på analysmetoder och -modeller/diagram 67
Tabell 5 UML-diagram och deras tillämpning 69

Bilder

Figur 1 Kravhanteringens sammanhang..... 24
Figur 2 Grundläggande produktlivscykel (PLC)..... 77

© SQEB – Software Quality Engineering Board

9 Standarder

IEEE 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE 1233-1998 IEEE Guide for Developing System Requirements Specifications

IEEE 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document

IEEE 830, IEEE 1233 and IEEE 1362 are replaced by ISO/IEC/IEEE 29148:2011

ISO 9000 Quality management

ISO 12207 Systems and software engineering — Software life cycle processes

ISO 15288 System Life Cycle Processes

ISO 15504 Information technology — Process assessment, also known as SPICE (Software Process Improvement and Capability Determination),

ISO 31000 Risk Management - Principles and Guidelines on Implementation

ISO/EIC 25000 (previously ISO/IEC 9126 Software engineering — Product quality)

ISO/IEC 14598-1:1999 Information technology -- Software product evaluation

SWEBOK - The Guide to the Software Engineering Body of Knowledge: http://www.computer.org/portal/web/swebok/home

SEBOK – Systems Engineering Body of Knowledge: http://www.sebokwiki.org/

© SQEB – Software Quality Engineering Board

10 Litteratur

Cockburn, A.: *Writing Effective Use Cases*. Amsterdam, 2000

Cohn M.: *Estimating With Use Case Points*. Fall 2005 issue of Methods & Tools

Cohn M.: *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional, 2004

Cotterell, M. and Hughes, B.: *Software Project Management*. International Thomson Publishing, 1995

Davis, A. M.: *Just Enough Requirements Management. Where Software Development Meets Marketing*. Dorset House, 2005, ISBN 0932633641

DeMarco, T.: *Controlling Software Projects: Management, Measurement and Estimates*. Prentice Hall, 1986

Doran, G. T.: *There's a S.M.A.R.T. way to write management's goals and objectives*. Management Review, Volume 70, Issue 11, 1981, (AMA FORUM), pp. 35-36.

Evans, E. J.: *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Amsterdam, 2003

Graham, D. et al: *Foundations of Software Testing*. London, 2007

Gilb, T.; Graham, D.: *Software Inspection*. Reading, MA, 1993

Gilb, T.: *What's Wrong with Requirements Specification.* See: www.gilb.com

Hull, E. et al: *Requirements Engineering*. Oxford, 2005

IBAQB: *Certified Business Analyst, Foundation Level Syllabus*, version 2011

IBUQ: *Certified Professional for Usability Engineering*, *Foundation Level Syllabus*, version 2011

ISTQB: *ISTQB Glossary of Testing Terms* v2.2

ISTQB: *Certified Tester, Foundation Level Syllabus*, version 2011

Jacobson, I. et al.: *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley, 1993

Lauesen, S.: *Software Requirements: Styles and Techniques*. London, 2002

Pfleeger, S.L. and J.M. Atlee: *Software Engineering: Theory and Practice.* Upper Saddle River, New Jersey, USA, Prentice Hall, 2006.

Project Management Institute: *A Guide to the Project Management Body of Knowledge* (PMBOK ® Guide). PMI, 2004

Robertson, S.; Robertson, J.: *Mastering the Requirements Process*. Harlow, 1999

Rupp, C.: *Requirements-Engineering und Management. Professionelle, Iterative Anforderungsanalyse in der Praxis*. Munich, 2007

Sharp H., Finkelstein A. and Galal G.: *Stakeholder Identification in the Requirements Engineering Process.* 1999

© SQEB – Software Quality Engineering Board

Sommerville, I.: *Requirements Engineering*. West Sussex, 2004

Sommerville, I.: *Software Engineering 8*. Harlow, 2007

Sommerville, I.; Sawyer, P.: *Requirements Engineering: A Good Practice Guide*. Chichester, 1997

Sommerville, I.; Kotonya, G.: *Requirements Engineering: Processes and Techniques*. Chichester, 1998

Thayer, R. H.; Dorfman, M.: *Software Requirements Engineering, 2nd edition*. Los Alamitos, CA, 1997

Wiegers, K.E.: *First Things First: Prioritizing Requirements.* Software Development, September, 1999

Wiegers, K. E.: *Software Requirements*. Redmond, 2005

Young, R. R.: *Effective Requirements Practices*. Addison-Wesley, 2001

© SQEB – Software Quality Engineering Board

11 Index

Acceptanskriterier, 54, 81
Agilt angreppssätt, 82
Användarberättelse, 54
Användningsfall, 54, 63
Användningsfallsanalys, 73
Avtal, 41
Baseline, 37
Begränsning, 11
BPMN, 54, 73, 75, 79, 91
Brainstorming, 54, 60, 64
CMMI, 18, 19, 23, 82, 88, 89
Delphi-metoden, 69
Enkät, 54
Funktionella krav, 14, 65
Generisk kravhanteringsprocess, 28, 29, 51
GUI-modellering, 54, 76
Horisontell spårbarhet, 37, 45
Horisontella prototyper, 76
Icke-funktionella krav, 14, 65
Identifiering baserat på befintlig dokumentation, 54
Intervju, 54, 60
Intressent, 28
ISO 12207, 23, 96
ISO 15504, 23, 96
ISO 9000, 22, 50, 96
ISO 9126, 22
ISO/IEC 25000, 22, 65
ISO/IEC/IEEE 29148, 15, 16, 22, 45, 78, 96
Kano-modellen, 16
Konceptuell modell, 54, 72
Konfigurationshantering, 37, 46
Krav, 7, 11, 14, 16, 17, 21, 27, 31, 32, 44, 51, 62, 65, 68, 84, 86, 91, 93
Kravanalys, 30, 31, 54, 56, 68, 78, 79
Kravhanterare, 1, 5, 27, 28, 48, 71
Kravhantering, 6, 7, 9, 11, 13, 17, 19, 24, 25, 27, 28, 30, 33, 37, 51, 54, 58, 82, 90
Kravinsamling, 30, 31, 54, 56, 57, 88, 91
Kravledare, 28
Kravledning, 11, 19, 32, 37, 40, 89
Kravmodell, 54, 72
Kravspecifikation, 31, 54
Kravspårning, 32, 44
Kravutvecklare, 28
Kravutveckling, 11, 19, 31, 54, 77, 84, 89
Kritikalitet, 11, 15
Kund, 28, 48, 88
Kundlärande, 54, 60, 63
Kundrepresentant, 54, 61
Kvalitetskontroll, 37
Kvalitetssäkring, 30, 32, 37, 39, 40, 48, 50, 51
Leverantör, 28, 88
Lösning, 11, 15
Lösnings-/systemkrav, 15
Lösningsmodellering, 71, 72
Lösningsspecifikation, 54
Mognadsmodell, 82
Mål, 6, 7, 54, 58, 95
Mätetal, 37, 52

© SQEB – Software Quality Engineering Board

Observation, 60, 62
Pilotprojekt, 90
Planning Poker, 54, 69
Prioritering, 54, 70
Prioritet, 11, 15, 51
Processmodell, 82
Produkt, 11, 15, 82, 86
Produkt-/komponentkrav, 15
Produktbacklog, 82, 86
Produktlivscykel, 82
Produktrisk, 37
Projektledning, 27, 48
Projektrisk, 37
Proof of concept, 90
Påverkansanalys, 37, 44, 48, 70
Rational Unified Process, 66, 82, 85
Risk, 17, 37, 42, 96
Riskanalys, 32, 37, 43
Riskhantering, 27, 37, 42
Riskhanteringsplan, 37
Riskidentifiering, 37, 43
Riskminskning, 37, 43
S.M.A.R.T., 54, 58, 97
SEBOK, 23, 96
Självloggning, 59, 61
Specifikation, 54
SPICE, 23, 82, 88, 96
Spårbarhet, 15, 27, 37, 40, 43, 44, 51, 85
SWEBOK, 16, 19, 22, 96
SysML, 54, 73, 75
System, 11, 22, 23, 57, 75, 96
Test, 78
Testbarhet, 51
UML, 54, 55, 71, 73, 74, 75, 76, 79, 80, 91, 95
Validering, 11, 19, 30, 50, 51, 55, 56, 81
Validering och verifiering av krav, 30, 50, 51, 55, 56, 81
Verifiering, 11, 19
Verksamhetsanalys, 24, 25, 27
Verksamhetsbegränsningar, 14
Verksamhetsbehov, 31
Verksamhetskrav, 15, 31
Verksamhetsproblem, 11
Verktyg, 7, 31, 32, 91, 92
Vertikal spårbarhet, 37, 45
Vertikala prototyper, 76
Vision, 54
Workshop, 54, 64
Åtagande, 11, 15, 16
Återanvändning, 45, 54, 60, 62
Ändring, 37
Ändringsbegäran, 32, 37
Ändringshantering, 37

© SQEB – Software Quality Engineering Board