F: **Vad är Lehmans lag?**

S: Det är en lag som rör programvara, som beskriver hur programvara utvecklas med tillräckligt med tid. Lehmans lag kan delas upp i 4 huvudkomponenter.

* "Kontinuerlig förändring" – ett mjukvarusystem måste anpassas kontinuerligt, annars blir det gradvis mindre tillfredsställande.
* "Ökande komplexitet" – när ett mjukvarusystem utvecklas ökar komplexiteten om inte arbete görs för att underhålla eller minska det.
* "Fortsatt tillväxt" – det funktionella innehållet i ett system måste ständigt ökas för att upprätthålla användarnas tillfredsställelse under dess livstid.
* "Sjunkande kvalitet" – kvaliteten på ett programvarusystem verkar sjunka om det inte upprätthålls och anpassas strikt till förändringar i driftsmiljön.

**F: Vilka är de vanliga stegen för att utveckla programvara?**

S: Start: Ett problem identifieras

Analys: Finns det efterfrågan på att lösa problemet?

Design: Användarberättelser och en övergripande uppfattning om hur programvaran kommer att se ut/kännas

Utveckling: omsätta designen i praktiken

Testning: Fixera designen efter att ha omstetat den i praktiken

Release: det är en release

**F: Beskriv en process och hur de används inom programvaruteknik**

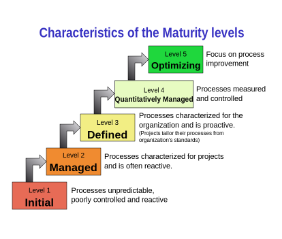
S: En process kan ses som en aktivitet eller på annat sätt som kräver en input och vissa villkor som ska uppfyllas innan något kan göras. En process avslutas när en uppsättning andra villkor har uppfyllts, vilket resulterar i en utdata. Den här utdata kan sedan bli indata för en annan process. Man kunde se vart och ett av de allmänna stegen för att utveckla programvara som en process.

**F: Vad är en lösning och hur skiljer den sig från processer?**

S: Lösningen är standardiserade processer, en så kallad "bra" process. Problemet är att en process som en storlek passar alla lösningar är svår att hitta och bygga, därför kan lösningen i ett annat scenario bara vara en process i ett annat. Lösning, skapa ramverk som bedömer processer, är de bra eller inte?

**F: Wtf är ett CMMI**

S: CMMI, även känd som Capability Maturity Model, är ett ramverk som behandlar programvaruprocessen. Den beskriver vad processen är byggd och underhållen.



**F: Beskriv faserna av mjukvaruutveckling**

A.:

* Förstudie – finns de förutsättningar som behövs för att lösa det problem som finns...
* Krav – vad är förväntningarna på den slutliga lösningen, eventuella måsten eller begränsningar ...
* Arkitektur och design – grundläggande struktur och komponenter, tillämpa vanliga mönster ...
* Utveckling – med tanke på den grundläggande strukturen och designen, konstruera programvaran ...
* Verifiering och validering - fungerar programvaran, och tar den exakt itu med det poserade problemet ...
* Distribution – starta systemet till de avsedda användarna ...
* Underhåll – uppdatera systemet baserat på ökad kunskap eller förändrad miljö, även åtgärda problem ...
* Pensionering – ta programvaran ur drift, adressera återstående användare, säkra data som inte får gå förlorade ...

**F: Förklara förstudie/genomförbarhet**

A.:

Förstudie/genomförbarhet: Är en kort och fokuserad aktivitet med avsikt att få mer kunskap för att avgöra om det är möjligt att leverera det avsedda systemet Svara på 3 frågor:

1. Ger systemet önskat värde?
2. Kan systemet implementeras inom budget ochtidsram?
3. Om det behövs, kan systemet interagera med andrasystem?

Detta kräver naturligtvis att detta system kan utvärderas i vissa avseenden.

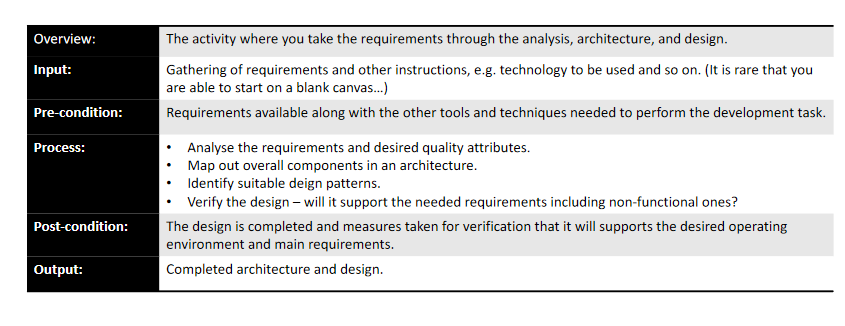
Förstudie innebär att du identifierar en möjlig skiss för hur lösningen kan se ut, undersöker de stora kostnaderna och tidssänken, bedömer osäkerheter som kan finnas och slutligen gör en bedömning om genomförbarheten med tanke på din ny funna kunskap. Förstudie fungerar som den skiss/skiss som du fortsätter med att skapa ditt system.

**F: Förklara kravteknik**

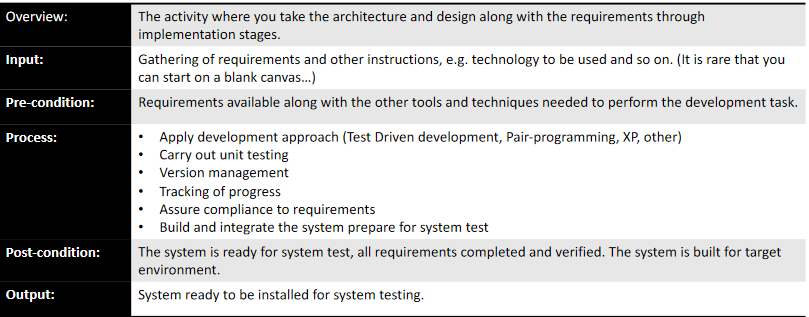
S: Kravteknik är en metodik och en uppsättning verktyg som används för att formulera de angivna kraven från intressenterna på ett mer exakt och konkret sätt.

Detta kräver naturligtvis att vissa krav ställs, tillsammans med en tydlig förståelse för vem målgruppen är, samt de förutsättningar under vilka utveckling kommer att ske.

Processen i sig handlar om att tillämpa en lämplig metod baserad på vem du får kraven från för att sedan identifiera kraven och dokumentera dem i enlighet därmed. Kraven kan kategoriseras i 3 olika grupper, funktionella, icke-funktionella och rättsliga krav. När detta har gjorts kan du börja prioritera vissa krav för att skapa en mvp. Vilket resulterar i både en kravspecifikation och en mvp

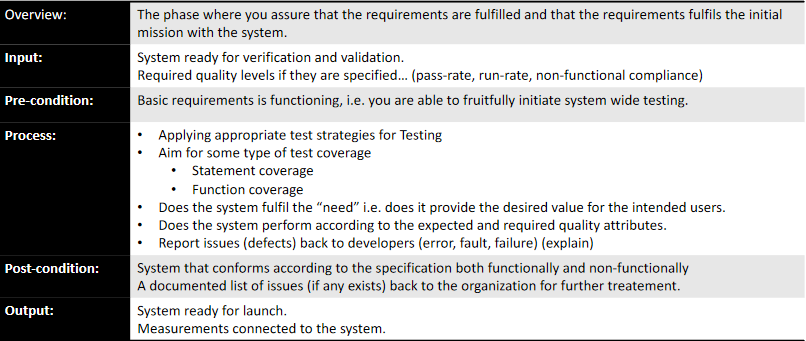
**F: Förklara programvarudesign**

**F: Förklara implementering**

A.: 

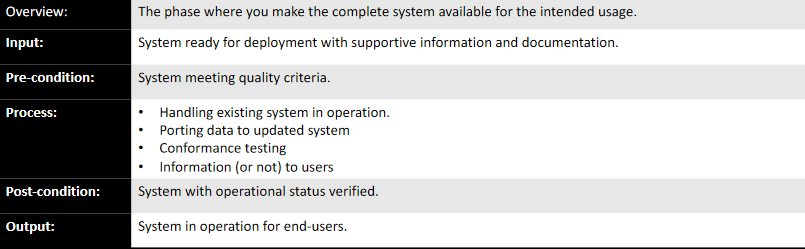
**F: Förklara verifiering och validering**

A.:



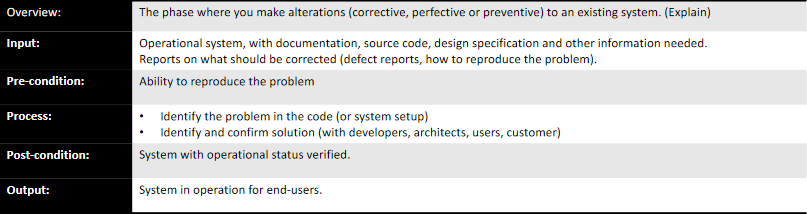
**F: Förklara distribution**

A.:

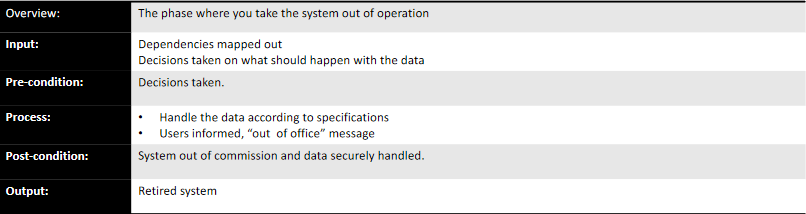


**F: Förklara evolution och underhåll**

A.:



**F: Förklara pensionen**

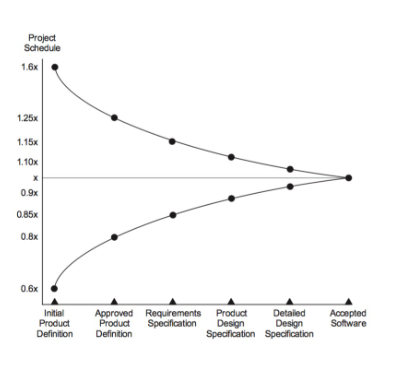
A.:

**F: Vad är vattenfallsmetoden?**

S: Vattenfallet, som det innebär med sitt namn, är en enkelvägs linjär metod för att utveckla programvara. Där varje steg kommer efter det andra och det inte finns någon väg tillbaka, när du har börjat lägga väggarna kan du inte ändra grunden.

**F: Varför är det dyrare att ändra din programvara, ju längre in i processen du går? Varför är uppskattningar i de tidiga stadierna också mycket felaktiga?**

S: Under de första utvecklingsstadierna finns det många okända okända såväl som kända okända som kan ta minuter att lösa eller veckor. När tiden går kommer mindre av dessa att visas och därmed kommer din tidsuppskattning inte att ogiltigförklaras av många dolda variabler.

Kostnaden för förändring i de tidiga stadierna kommer att vara nära ingen eftersom produkten definieras, krav kan läggas till på ett infall eftersom ingenting har utvecklats ännu. Under utvecklingsfasen kan kostnaden variera beroende på både hur långt in i det här skedet du är och vad du vill ändra. Detta beror främst på att utvecklingen på de delar som kan komma att ändras redan kan ha skett, eller att det skulle skapa nya beroenden som kan störa din plan. Alla dessa kostnader är mindre jämfört med kostnaderna efter frigöraren. När en produkt är ute på marknaden kan förändring kräva en hel remake av produkten eller för att du ska fixa hemsk kod som fungerade med de tidigare kraven.

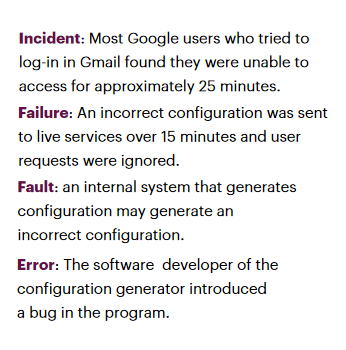
Note: **Kolla på föreläsning 3 plandriven validation och spiral modellen**

**F: Förklara skillnaderna mellan validering och verifiering**

S: Verifiering är processen för att avgöra om produkten uppfyller de angivna specifikationerna på teknisk nivå medan Valideringar utvärderar om den är lämplig att göra sitt jobb.

**F: Vad är ett fel, fel och fel när det gäller testning?**

S: Ett fel är något som gjorts av utvecklaren som kan orsaka ett fel, medan ett fel är något som kan orsaka felaktig funktion. Om ett fel orsakar felaktig funktion kan det klassificeras som ett fel.



**F: Vilka är målen med testning?**

A.:

* Om du vill hitta fel innan de orsakar ett system
* utföra de tester som krävs effektivt och effektivt inom dina begränsningar.
* Så här tar du upp kvaliteten på programvaran genom att ta bort eventuella fel
* För att notera programvarufel för framtida användning i förebyggande och båda korrigerande åtgärder.

**F: Vad är optimistisk felaktighet**

S: Optimistisk felaktighet innebär att du accepterar vissa program som inte har egenskapen (dvs. det kanske inte upptäcker alla överträdelser).

Du är okej med att veta att du inte hittar alla fel, så länge de flesta av dem hittas.

**F: Vad är pessimistisk felaktighet?**

S: Pessimistisk felaktighet vi kanske inte accepterar ett program även om det har fastigheten.

Låt oss säga att ditt program ger rätt utdata men av en eller annan anledning (falska positiva saker till exempel) accepteras det fortfarande inte av någon utvärderare. Detta sker främst när enutomerad programanalysteknik tillämpas.

**F: Vad är förenklade egenskaper?**

S: Minska fastigheterna frihetsgrad. Du förenklar egenskaperna när du inte kan nå en acceptabel funktionalitetsnivå i programmet.

**F: Förklara stegvis testning och bigbang-testning**

S: Bigbang-testning är när du testar programvaran först efter att alltär tillgängligt , detta har naturligtvis många nackdelar eftersomspårbarhet av problem blir nästan omöjligt. Stegvis testning innebär att varje modul testas så snart den är tillgänglig, detta ökar detekteringsfrekvensen för buggar men ökar redundansen och kan beskatta resurser.

**F: Vad är en stub och vad är en förare?**

A: stub är en dummy ersättning av en modul som används för att testa en annan modul, medan en drivrutin är en faktisk modul.

**F: Förklara skillnaden mellan funktionell och icke-funktionell testning**

S: Funktionstestning är en typ av testning som verifierar att **varje funktion** i programvaran fungerar i enlighet med kravspecifikationen. Vanligtvis innebär blackbox-testning.

Icke-funktionell testning är en typ av testning för att kontrollera icke-funktionella aspekter (prestanda, användbarhet, tillförlitlighet etc.) av en programvara.

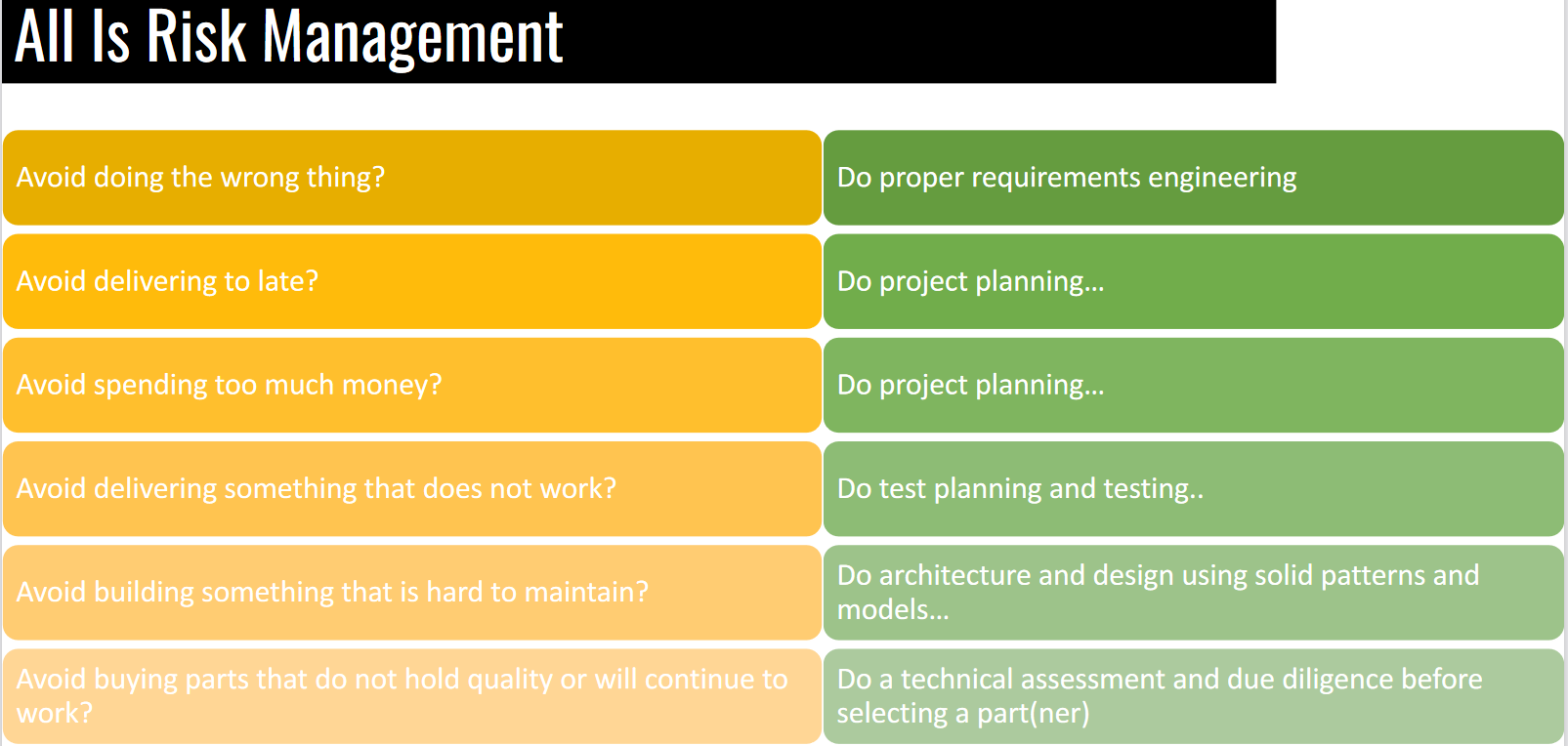
**F: Vad är ett statiskt test kontra dynamiskt test?**

S: Ett statiskt test är en typ av programvarutestning där kod inte körs under dynamisk testning av koden körs. Målet under statisk är att hitta fel i de tidiga tillstånden i din utvecklingsprocess. Under dynamisk testprestanda övervakas.

**F: Vad är riskhantering?**

**S:** Riskhantering handlar om att identifiera risker och minimera deras effekt på ett projekt. Riskbedömning ingår också under denna term.

**F: Nämn några saker som är riskhantering**

****

**F: Riskklassificering**

**A.:En bild som visar text

Automatiskt genererad beskrivning**

F: Lista stegen i risk-identifiering

Riskidentifiering är ett steg där alla möjliga risker listas ochkategoriseras. En checklista med vanliga risker finns ofta och det kan vara både en gruppasibilitet och en individuell uppgift. Checklistan kan se ut så här

En bild som visar text

Automatiskt genererad beskrivning

F: Vad gör du i risk analys?

S: Du bedömer sannolikheten och allvaret för varje risk, detta görs genom att skapa en tabell där du rangordnar varje problem baserat på de två ovannämnda måtten. Tabellen kan se ut så här:

En bild som visar bord

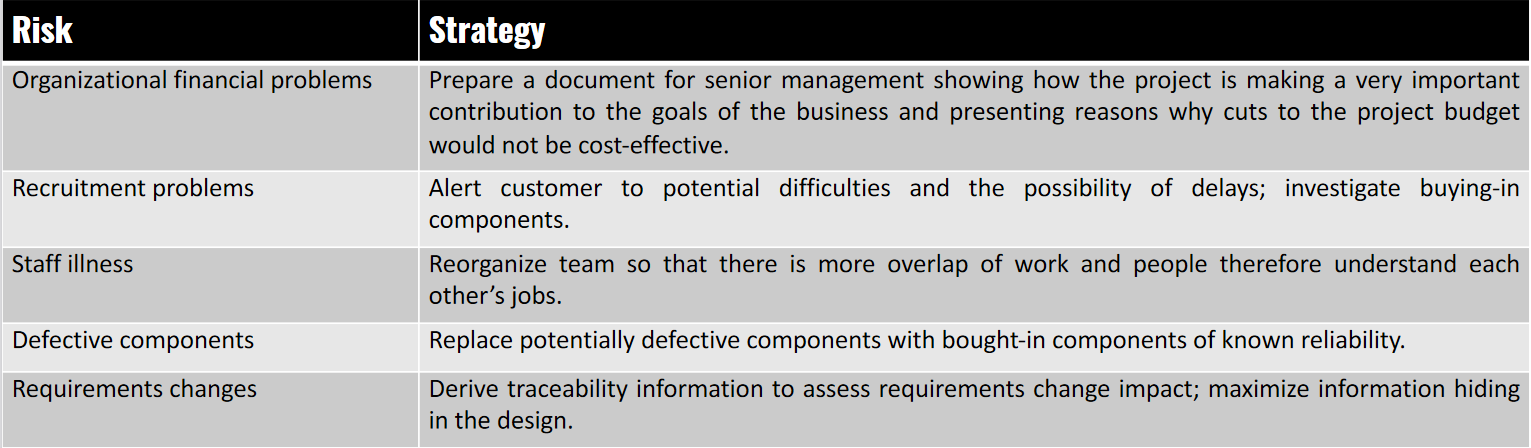
Automatiskt genererad beskrivning

F: Vad är risk förplanering?

S: Riskplanering innebär att utveckla strategier för varje risk för att hantera dem. Gemensamma strategier är undvikande, minimeringsstrategier och beredskapsplaner. En undvikandestrategi syftar till att minska sannolikheten för att risken kommer att uppstå. Minimationsstrategier minskar svårighetsgraden av den här risken medan beredskapsplanen finns där om allt går åt helvete. Detta är några exempelstrategier

En bild som visar bord

Automatiskt genererad beskrivning



**F: Vad är riskövervakning?**

S: Utvärdera varje identifierad risk regelbundet för att avgöra om det blir mer troligt samtidigt som man bedömer om effekterna av denna risk är desamma. Riskövervakning kräver också att du diskuterar de viktigaste riskerna vid möten för att se om något nytt har hänt.

F: Vilka är några alternativa metoder för riskhantering?

S: FMEA/FMECA (felläge och effektanalys) är ett alternativ som utvecklades på 40-talet. Denna metod har framgångsrikt tillämpats inom fordonsindustrin och andra säkerhetskritiska branscher samt mjukvaruindustrin. Genom att rangordna flera aspekter av en risk skapas en poäng av deras produkt somhbedömer svårighetsgraden. (lägre tal = mindre allvarliga)

En bild som visar text, vägg, skärmbild, skärm

Automatiskt genererad beskrivning

F: Säg några saker om återanvändning av programvara

S: Numera utvecklar vi programvara genom att integrera bitar och bitar av befintlig programvara, bredvid kodrader skrivs från början nuförtiden. Återanvändning av programvara ger snabbare tid att marknadsföra samtidigt som utvecklingskostnaderna och processriskerna sänks. Programvara är en värdefull och återanvändbar tillgång, att återanvända den ökar din avkastning på den tid det tog att ursprungligen utveckla den programvaran.

F: Lista fördelarna och nackdelarna med återanvändning av programvara.

Återanvändning av programvara minskar tiden till marknaden eftersom det inte finns något behov av att utveckla varje bit och del av produkten. Detta frigör också experter att göra det de är bra på istället för att utveckla redan befintlig kod. Att återanvända andras programvara skapar en gemenskap av pålitlig förmågaoch förtroende. Ett sådant samhälle sätter press på människor att följa standarder som skapar interoperabilitet. Rmedför risker eftersom den programvaran redan har testats tidigare.

Med detta sagt kan det ta tid att hitta och använda återanvändbara komponenter och kan öka underhållskostnaderna eftersom modulen kanske inte passar perfekt vilket migh kräver att du anpassar något som inte är ditt. Återanvända programvara kan också orsaka "inte uppfunnits här syndrom", en idé som eftersom du inte har skapat det modulen kan vara dålig. Det kan också finnas en brist på stöd.

F: Beskriv komponentbaserad programvaruteknik

S: Komponentbaserad programvaruteknik förlitar sig på återanvändning av entiteter som kallas komponenter. Komponenter är mer abstrakta/större än klasser och kan på många gånger ses som fristående tjänsteleverantörer.

F: Vad är en komponent?

S: Komponenter är oberoende delar som sällan stör varandra. Deras implementeringar och faktiska kod är dolda eftersom all kommunikation är via ett gränssnitt. Komponenter kan ocksåplas om så länge gränssnittet upprätthålls (kolla detta senare fl8). De erbjuder också en rad standardtjänster

F: Vad är komponentbaserade programvarutekniska nödvändigheter?

A.: En bild som visar text

Automatiskt genererad beskrivning

F: Vilka är egenskaperna hos komponenter?

S: Komponenter är komposerbara (sammansättningsbar) och lätt att distribuera, de är vanligtvis väldokumenterade och independendant av varandra orsakar få konflikter. De flesta komponenter är också standardiserade.

F: Beskriv modelldriven teknik

S: Modelldriven teknik (MDE) är en metod för mjukvaruutveckling där modeller snarare än program är de viktigaste resultaten av utvecklingsprocessen. De program som körs på en maskinvaru-/programvaruplattform genereras automatiskt från dessa modeller.

Modelldriven programvara handlar i huvudsak om att använda modeller för att producera programvara. Modeller går från att vara ett sätt att beskriva programvara till att bli en aktiv del av mjukvaruutvecklingsprocessen. Detta ökar abstraktionsnivån (tänk hög/lågnivå programmering) så att vi inte behöver bry oss om programmering av språkdetaljer or plattformskrav.

F: Vilka är kraven i MDS

S: Modelldrivenprogramvaruutveckling kräver ett domänspecifikt språk (ett program som konverterar dina modeller. Detta innebär att MDS bara är livskraftigt om du utvecklar samma sak mer eller mindre. Annars är det allmänt opraktiskt att behöva utveckla ett sådant program för varje projekt.

Modelldriven programvara finns i mixer med sin nodbaserade skuggning skapande. Istället för att behöva programmera dina skuggningsprogram väljer du och väljer noder för att skapa en skuggning, det är mycket enklare men att skapa ett sådant gränssnitt för allt i mixer skulle vara omöjligt. Anledningen till att nodsystemet fungerar är att alla objekt är texturerade på samma sätt men med olika alternativ och för att konstnärer utvecklar flera skuggningsmedel och inte bara en under sin livstid.

F:Vilka är de olika nivåerna av modeller:

A.:

* CIM (Computation independent models): modeller där allt beskrivs på ett icke-tekniskt sätt
* Plattformsoberoendemodeller: modeller där plattformen inte anges (programmeringsspråk, os)
* Plattformsberoende modeller: modeller däre-plattformen beaktas
* kod

F:Vad är en översättare i mdsd?

S: En översättare minskar abstraktionen av din modell, vilket för modellen ett steg närmare den faktiska koden för att skapa program.

F: Fördelar och nackdelar med mdsd

Fördelar: Gör det möjligt att överväga system på högre abstraktionsnivåer.

Att generera kod gör det automatiskt billigare att anpassa sig till nya plattformar och system.

Nackdelar: Spara från att generera kod kan överskuggas av kostnaden för att utveckla en översättare för de nya plattformarna. Argumenten för plattformsoberoendedence är endast giltiga för stora system med en lång livscykel. För programvaruprodukter och informationssystem är besparingarna från användningen av mda sannolikt mindre än kostnaderna för att införa det och bygga upp dessa verktyg

F: Vad är plandriven sd?

S: Plandriven mjukvaruutveckling är en utvecklingsmodell där allt är minutiöst planerat fram till veckan, så att varje medlem kan veta vad agendan är för en viss tidsram. Detta uppnås genom en projektplan.

F: Vad är en projektplan?

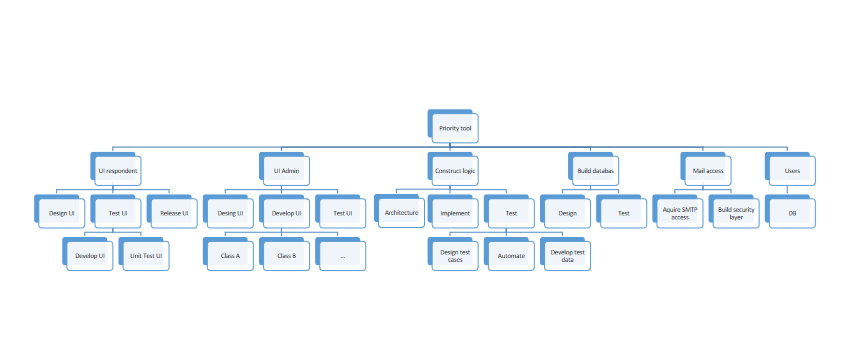
S: En projektplan är ett dokument som beskriver resurser, workbreakdown och schemat för att utföra arbetet. Det innehåller vanligtvis en introduktion till projektet (som fastställer begränsningar och en beskrivning av projektet), projektorganisationen, en riskanalys, hårdvaru- och programvarukrav, workbreakdown, projekt som planeras i en eller annan form samt övervaknings- och rapporteringsmekanik. Projektplanen kan vara så detaljerad som du vill att den ska vara.

F: Vad är ett krav?

S: Det är en maskinvaru- eller programvarufunktion som beskrivs medminsta antal antaganden som gjorts abou t de tillgängligaresurserna. Varje uppgift är uppdelad i små och hanterbara uppdrag.

F: Vad är en wbs och varför är det användbart?

S: En wbs, workbreakdown struktur är ett träd där varje krav är uppdelad i undertask och dessa underaktiviteter görs till subsubtasks tills en adekvat abstraktionsnivå har uppnåtts. Att göra det underlättar tidsuppskattningen, eftersom abstraktionsnivån på den här uppgiften minskar den ytterligare grenen ute på trädet. WBS avslöjar också beroenden. Kom ihåg att sluta förgrena dig så snart bladnoderna är lätta att uppskatta.



F: Vad är ett gantt-schema?

S: Ett gantt-schema är ett schema för när varje krav ska fungera. Både beroenden och resursallokering kan göras/beskrivs i en gantt shcedule.

F: Vad är en kritisk väg?

S: En kritisk sökväg är en beräkning som beskriver vilka aktiviteter som inte kan läggas ned om inte hela projektet ska försenas.

F: Vilka är problemen med pdsd.

S: Problemet är att tidiga beslut bör ses över på grund av förändringar i miljön, själva produkten, den utvecklande miljön eller kunden. Plandriven tillåter inte mycket förändring

F: När är plandriven lämpligt?

S: När säkerhet ochstabilitet/säkerhet är nyckeln.

F: Hur bedömer du den tid ett projekt/en aktivitet tar

S: Det finns flera sätt att uppskatta tiden, man kan tillämpa en algoritm som tar data och matar ut en grov uppskattning (COCOMO är ett exempel på detta)

Analoga modeller liknar aktiviteter/projekt med tidigare aktiviteter/projekt och använder det som referens för att basera sin tidsuppskattning. "Project x ser mycket ut som projekt y men det är lite svårare, y tog z timmar, x kommer att ta 2z timmar"

Konsensus är bara en grupp snubbar som diskuterar det tillsammans

Agila tekniker innehåller vanligtvis analoga modeller utan tidsuppskattning, grova gränser ges istället eftersom de vet att uppskattningenärprob ably fel.

F:Vad är skillnaden mellan agil och plandriven?

S: Plandrivna modellsteg bygger på varandra, föregående stegs utdata blir indata för nästa steg och så vidare. Agile består av korta 3-4 veckors sprint där hela processen med atttesta, utveckla, designa görs där för målanvändarna. Ta allt som det kommer och tillåta förändring.

F: Vad var mjukvarukrisen?

S: Mjukvarukrisen uppstod när programvaran blev mer oregerlig på grund av att maskinerna som körde dem blev mycket kraftfullare, vilket möjliggör större program. Tidigare var den mesta koden liten och följaktligen lätt att förstå.

F: Wtf är det agila manifestet?

S: Det är ett par meningar som beskriver skillnaden mellan agila och traditionella utvecklingsmetoder. "Individer och interaktioner över processer och verktyg", "Fungerande programvara över omfattande dokumentation", "Kundsamarbete över avtalsförhandling", "Svara på förändringar efter en plan". Detta betyder inte att avtalsförhandlingar eller processer och verktyg är oviktiga.

F:Vilka är principerna bakom det agila manifestet

A.:

Högsta prioritet är att tillfredsställa kunden genom tidig och kontinuerlig leverans av värdefull programvara.

Agile-processer välkomnar förändrade krav, även sent i utvecklingen, och utnyttjar förändringar för kundens konkurrensfördel.

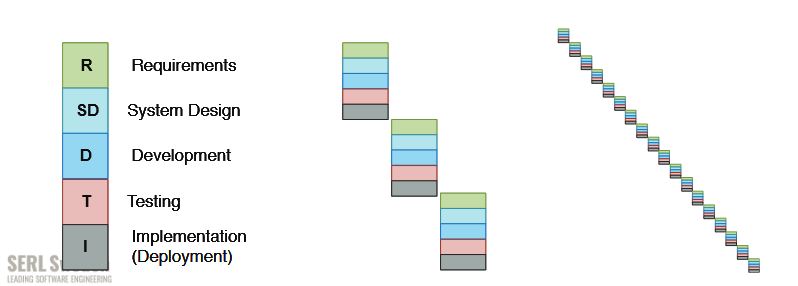
Leverera programvara ofta, från ett par veckor till ett par månader, med en preferens för den kortare tidsskalan.

Affärsmän och utvecklare måste arbeta tillsammans dagligen under hela projektet. Bygg projekt kring motiverade individer, ge dem miljö och stöd de behöver.

Den mest effektiva och effektiva metoden för att förmedla information till och inom ett utvecklingsteam är personlig konversation.

Agila processer främjar hållbar utveckling. Sponsorutvecklarna och användarna bör kunna upprätthålla en kontant takt på obestämd tid

F: Vilket är iterativ, vattenfall och extrem programmering?



A: längst till vänster är vattenfall, centrum är iterativt, längst till höger är xp

F: Vad är extrem programmering?

S: Den grundläggande idén med xp är att driva de principer som tillämpas i iterativ till det yttersta.

F:Vilka är kraven för att xp ska vara livskraftigt?

S: XP kräver en konstant kommunikation med kunden/en representant som ställer kraven till teamet. Du måste definiera systemet genom metaforer som delas med kunden för att få ett bättre grepp. Du behöver också användar teorier som dokumenteras i artikel kort definiera de grundläggande interaktionerna med systemet.

F: Hur får du användartories i xp?

S: Alla användar teorier tillhandahålls genom samarbete med kunden. Kunden eller deras representant beskriver vissa funktioner i form av en användarberättelse, vilket möjliggör förändring så snart den uppstår. Användarnatories dokumenteras sedan i artikelkort där grundläggande interaktioner beskrivs som exemplet nedan:

En bild som visar text

Automatiskt genererad beskrivning

F: Vad fan är ett planeringsspel?

S: Planeringsspelet är ett planeringsmöte som hålls av devteam och intressenter. Den har två sessioner; planering och iterationsplanering.

F: Beskriv en releaseplanning

S: Utgivningsplaneringen innehåller tre faser, utforskning, engagemang och vägledning.

I prospekteringsfasen samlar kunden in användarberättelser och skriver dem på användarkort. Devteam uppskattar de timmar som krävs för varje användare. Om en användare inte är uppskattad returneras den till kunden för ytterligare nedbrytning.

I engagemangsfasen prioriterar kunderna användarberättelser baserat på affärsvärde och utvecklarna prioriterar användarberättelser baserat på risk och bekräftar utvecklingshastigheten. Slutligen väljer kunden användarberättelsen som ska slutföras för nästa utgåva.

Under vägledningen ändras planen i accodrance med både dev och kundernas önskemål.

En bild som visar text, inomhus, skärmbild

Automatiskt genererad beskrivning

F: Beskriv iterationsplanering

S: Iterationsplaneringen planeras vidare utifrån releaseplanen. Varje berättelse är uppdelad i mindre uppgifter under utforskningen och uppskattningar görs om.

I åtagandefasen gör varje utvecklare frivilligt anspråk på uppgifter och gör en slutlig uppskattning av de uppgifter han ansvarar för och teamet bör utvärdera om de är överbelastade. En eller flera utvecklare ansvarar för en uppgift.

Under vägledningen implementerar utvecklare varje uppgift genom att para ihop med kontinuerlig integrering och testdriven utveckling.

En bild som visar text

Automatiskt genererad beskrivning

F: Vad är hastighet och hur beräknas det?

S: Hastighet är mängden artikelpoäng/uppgifter som devteam kan avsluta i en sprint. Deras hastighet bestäms av ovannämnda faktor, om ett lag inte avslutar en uppgift returneras det till scrumboarden med hälften av poängen.

F: Vad menar de med enkel design

S: Det är tanken att när man utvecklar en produkt bör man försöka skriva så få rader koder som möjligt för att bara klara testet och ingenting. Du bör också undvika att skapa fler metoder/klasser än nödvändigt och skapa program som uppfyller kraven och inget mer.

F: Förklara refaktorering,

S: Kontrollerad teknik för att ändra strukturen för en befintlig kod, vilket förbättrar kodens allmänna läsbarhet och kvalitet. Systemets design utvecklas kontinuerligt genom kodrefaktorer som utförs av utvecklarna så snart förbättringarna hittas i xp.

Exempel på refaktorer är:

* dela upp stora klasser i mindre klasser som utför samma arbete
* förkorta metoderna genom att dela upp dem
* rensa upp relationer mellan funktioner och göra dem mer förståeliga.

F: Sammanfatta TD och dess orsaker/begränsningsstrat

S: Teknisk skuldsättning är en form av skuldsättning som uppvisats genom de långsiktiga effekterna av suboptimala beslut som tidigare fattats i devphase. Refaktoring är den mest tillämpade lösningen för att mildra TD; men det minskar nödvändigtvis td vi siktar på, till exempel buggar.

F: VILKA ÄR DE VIKTIGASTE ÖVAR AV XP

A.:

* Parprogrammering: All kod produceras av par av utvecklare som fungerar helt och hållet
* Kollektivt ägande: Alla programmerare kan arbeta var som helst i systemet och utföra refaktorer när som helst, var som helst, om de ser möjligheten. Vem som helst kan också ändra vad som helst. Detta tar bort de möjliga öarna av expertis och alla medlemmar i teamet kan arbeta och förstå hela koden.
* Kodningsstandarder: alla kan potentiellt arbeta vartenda gång i koden. Alla kan utveckla alla andras tidigare arbeten. Vi behöver kodningspraxis.
* 40h veckor: Ingen kan arbeta en andra vecka i rad med övertid, detta kommer att påverka kvaliteten på koden och kan vara ett problem som är inneboende i företaget.
* Öppna arbetsytor bör vara tillgängliga för kommunikation.

F: Hur fungerar parprogrammering?

S: All kod skrivs av två utvecklare som arbetar helt och hållet med en dator. En utvecklare tänker på den bästa lösningen för problemet till hands medan den andra killen tänker mer strategiskt på de eventuella konsekvenserna.

F: Hur utförs testning i xp?

S: Test första utvecklingen tillämpas, där devs skrivenhetstester för varje ny funktionalitet redan innan funktionaliteten implementeras.

Sedan, genom kontinuerlig integration, är den nya koden integrerad efter några timmars producerad? Vid integrering är systemet helt ombyggt och alla måste passera, eller så ignoreras batchen med ändringar.

F: Hur genomför du test först utveckling?

S: Genom att dela upp användarberättelserna i uppgiftskort kan var och en av uppgifterna sedan anges i ett eller flera enhetstester som kontrollerar implementeringen med hjälp av kunden. Detta gör det möjligt för utvecklaren att veta exakt hur systemet ska bete sig och hur det inte ens borde göra det innan det är klart.

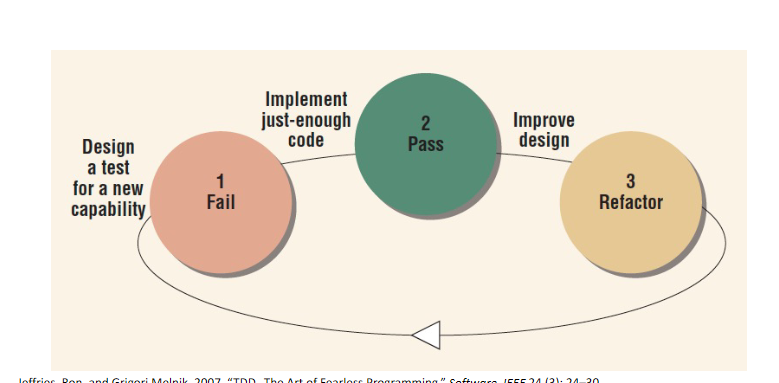
F: Vad är testdriven utveckling?

S: Testdriven utveckling (TDD) är en mjukvaruutvecklingsmetod som föreslår att utvecklings- och testaktiviteterna ändras. Man kan TDD som en utveckling av test-första metoden i XP. Det är viktigt att notera att TDD är en design- och programmeringsaktivitet, inte en testaktivitet i sig

F: Beställ och beskriv den grundläggande processen med TDD

A.:

1. Identifiera den ökning av funktionaliteten som skulle vara ett bra komplement till systemet
2. Skriv ett test som anger hur systemet ska anropa funktionaliteten och vad resultatet ska bli
3. Kör testet (testet visar inte att funktionen inte redan finns, vilket visar att ditt test är funktionellt)
4. Utveckla minimikoden som gör att det nya testpasset sedan verifierar att alla tidigare tester fortfarande klarar
5. Refaktoring: granska koden



F: Vad är scrum?

S: Scrum är ett ramverk för organisingagila projekt och för att ge synlighet om deras framsteg. Det är lätt att integrera redan befintliga utvecklingssyror.

Ett scrum-team består av:

* en produktägare, en person som kontrollerar att allt är i enlighet med kundernas önskemål. Inspekterar om alla användarberättelser redovisas och om de uppfyller kundernas krav med dessa berättelser
* ett devteam kan inte vara större än 7 personer av någon anledning,
* en scrummaster som agerar som en tränare och vägledning för att säkerställa att scrum-processen följs.

F: Vad är en eftersläpning?

S: Ett huvudkoncept bakom scrum är idén om en produktstock, en slags att göra-lista över vad som ska göras. Detaljnivån för artiklarna varierar kraftigt, eventuellt inklusive eller inte inklusive underaktiviteter, beskrivning och sådant. Det är helt upp till devteams diskretioner, med hjälp av produktägaren. Berättelserna i eftersläpningen ges berättelsepunkter, dessa berättelsepunkter är ett mått på uppgiftens storhet.

F: Beskriv scrumprocessen

Scrum själv är organiserad kring sprintar för att hantera utvecklingen av produkten. Produktägaren bestämmer vilka uppgifter som ska hanteras. Eftersom produkten kan vara biig, är den uppdelad i en lista över hanterbara funktioner genom en aktivitet som kallas grooming. Funktionerna samlas sedan in i en prirotiserad lista (eftersläpningen). En sprintbacklog skapas med en lista över de funktioner som planeras att genomföras under den givna sprinten. Mängden funktioner som är färdiga i en viss sprint beror på hastigheten.

F: Vad är en potentiellt leveransbar produktstegring

S: Det är tanken att allt som utvecklas i en sprint är redo att distribueras och användas i produkten. Alla tester och arbeten finns i den här sprinten.

F: Vad gör du i ett scrum-möte och när inträffar de?

S: Scrum-möten hålls dagligen med målet i åtanke att granska framstegen och priotisera det arbete som ska göras under dagen. Scrum-möten hålls idealiskt ansikte mot ansikte och inkluderar hela laget. Mötena manifesteras som korta stand up-möten där omskrivningar görs. Dessa möten gör det möjligt för teamet att reagera snabbt och dra beredskapslösningar på en stunds varsel. De är baserade på tre frågor:

Vad åstadkom jag igår?

Vad ska jag jobba med idag?

Något som hindrar mig från att utvecklas?

F: Vad är en scrumboard och vilket syfte tjänar den?

S: Scrumboard, är en översikt över projektets aktuella tillstånd. Anteckningar representerar uppgifterna i produktstocken och du kan kasta anteckningarna i olika kategorier (inte gjort, todo, gjort, pågående). Det fungerar som ett kommunikationsverktyg för teamet.

F: Hur fungerar sprintplanering?

S: Varje sprint har en fast längd, vanligtvis mellan 2-4 veckor. I början av sprinten prioriterar produktägaren vissa artiklar från eftersläpningen för att definiera de viktigaste artiklarna som ska utvecklas i cykeln. Teamet gör sedan ett urval av de högst prioriterade punkterna i eftersläpningen, vilket gör dem till kandidater att bli devloped i sprinten.

Produktägaren tillsammans med teamet gör en uppskattning av den tid som krävs för att slutföra varje artikel. Den uppskattningen baseras vanligtvis på hastigheten (hur mycket av eftersläpningen som kan täckas i en enda sprint). Laget skapar en sprint backlog och sedan själv organiserar. Det är viktigt att sprinten innehåller både stora och små uppgifter för att inte förstöra framtida sprintar.

F:Vilka är de viktigaste reglerna för scrumming

A.:

En bild som visar text, inomhus, skärmbild

Automatiskt genererad beskrivning

F: Vad gör du i slutet av varje sprint?

S: Två möten hålls för att analysera återupplivningar på produkt- och processnivå?

En bild som visar text

Automatiskt genererad beskrivning

F: Vem upprätthåller eftersläpningen och varför?En bild som visar text

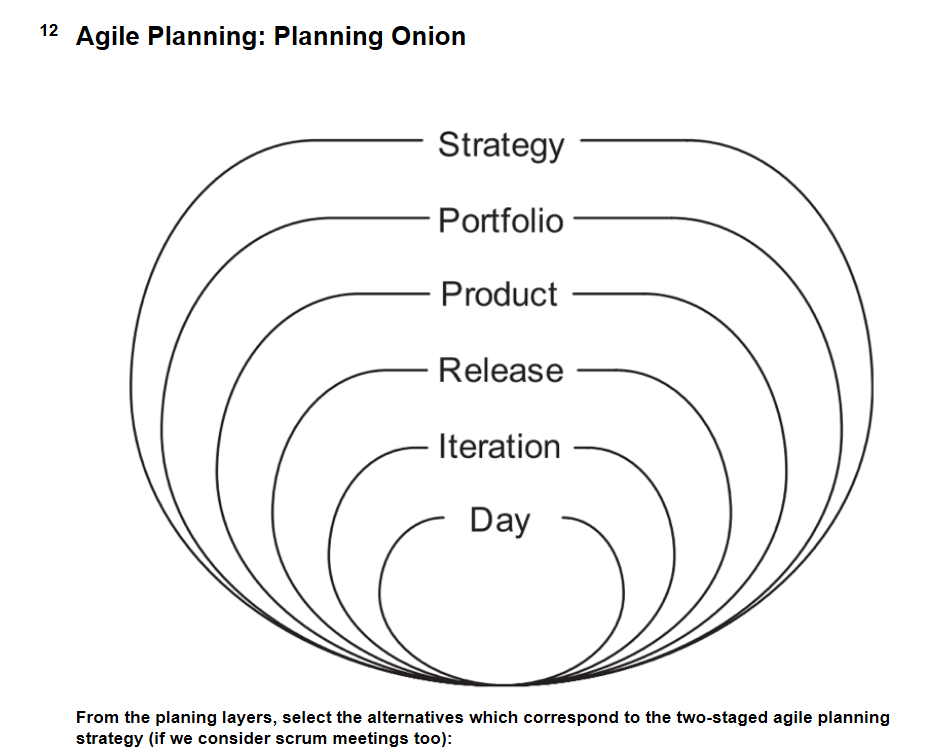
Automatiskt genererad beskrivning

F: Varför är scrumming så stark?

S: Scrum förutsätter att kraven kommer att ändras och därför möjliggör förändring. Målet är inte att starta projektet med en fullständig specifikation av kraven. Detta gör det möjligt för teamet att växa och få kunskap om projektet eftersom det är utvecklat för att skapa en ännu bättre produkt än planerat. Kraven kommer att dyka upp när koden testas, detta händer inte i vattenfallsmetoder eftersom testning sker i de senare stadierna och gränserna ändras. Den iterativa karaktären är det som ger scrum en sådan fördel att ta itu med förändringar.

F: Hur uppskattar du storypoints?

S: Det finns främst två tekniker för att estamiting storypoints. Eftersom storypoints är ett relativt mått bygger båda dessa strategier på att fastställa en standard. Att försöka hitta minsta möjliga användaretory och tilldela den 1 berättelsepunkt är den första strategin medan den andra strat kräver att du hittar den perfekta medieberättelsen och tilldelar den 5. När en standard har lagts kan du jämföra dina berättelser med din standard och tilldela värden. När du har tilldelat några som du säkert tror har rätt värden kan de också fungera som standarder för jämförelse.



En bild som visar text

Automatiskt genererad beskrivning