Test av IT-system - 21TI1B

Test av IT-system (Högskolan i Borås)

Skanna för att öppna på Studocu

Test av IT-system

21TI1B HT2024

Kurslitteratur

- ☐ Eriksson, Ulf (2008). Test och kvalitetssäkring av IT-system. 2. uppl. Lund: Studentlitteratur
- ☐ Holmquist, Eva (2018). Praktisk mjukvarutestning. 1. uppl. Lund: Studentlitteratur

---

TENTA

Om test; översikt, helhet, begrepp och standarder

Varför test?

1. För att identifiera fel.
2. För att säkerställa kvalitet.
3. För att skapa ett användbart system.

***För att verifiera**; kontrollera att det färdiga resultatet överensstämmer med kravställningen.*
***För att validera**; säkerställa att resultatet fungerar i praktiken och motsvarar verklighetens behov.*
För att kontrollera om systemet eller programvaran uppfyller kravspecifikationen och för att hitta fel.

Definition av testning i IEEE 829-2008

- Processen att exekvera eller evaluera ett system genom att använda manuella eller automatiska metoder för att verifiera att systemet uppfyller specificerade krav eller att identifiera skillnader mellan förväntade och verkliga resultat.
- **IEEE 829-2008** är en standard för att strukturera och dokumentera testprocesser, inklusive testplaner, testfall och testrapporter, för att säkerställa tydlighet och spårbarhet.

Vad är standard?

En standard är en gemensam lösning på återkommande problem som skapar enhetliga rutiner för att höja kvalitet och effektivisera arbete. De underlättar kommunikation, upphandling och avtalsskrivning samt fungerar som riktmärken för god praxis. I Sverige ansvarar SIS för standardisering, medan ISO är det största internationella organet.

Standarder och certifiering

IEEE – Institute for Electrics and Electronic Engineers = Internationellt standardiseringsorgan som sätter standarder för allt mellan systemtest och kärnkraftsanläggningar.
ISTQB – International Software Testing Qualification Board = Certifierar testare.
SSTB – Swedish Software Testing Board = Utbildar och certifierar testare.
TMM i – Test Maturity Model integrated = Används för att certifiera mognadsgraden på testaktiviteter hos en systemutvecklings organisation.

IEEE Standard 829-2008

Tillhandahåller dokumentationsmallar:

- Definierar testuppgifter, nödvändiga förutsättningar (input) och förväntade resultat (output).
- Innehåller en *spårbarhetsmatris* för att säkerställa att alla krav täcks.
- Dokumenterar vad som gjordes och varför, samt avvikelser från den ursprungliga planen.

Struktur för hela testprocessen:

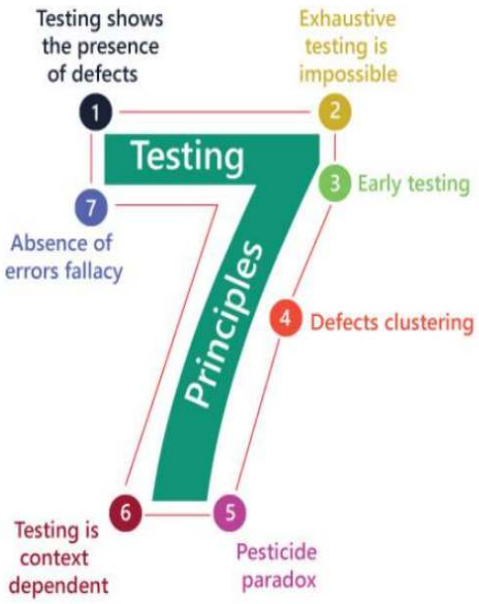
- Bryter ner den övergripande testplanen i mindre delar för att underlätta genomförandet.
- Testarbetet delas upp i olika nivåer, såsom enhetstestning och systemtestning.
- Separata dokument skapas för planering och testfall, vilket ger tydlig struktur.

Kritik = Kan **inte vara flexibel**, leder till **slöseri med tid** och kan ha **fel fokus** om det inte används korrekt.

Fördelar = Bra checklista för att säkerställa att alla viktiga delar täcks - om den används **flexibelt**.

7 principer om test

1. **Test påvisar att fel finns:** Testning visar att det finns fel, men kan aldrig bevisa att systemet är felfritt.
2. **Fullständiga test är omöjliga:** Det går inte att testa allt; prioritering krävs för att täcka de viktigaste områdena.
3. **Testa tidigt:** Tidig testning identifierar fel snabbare och minskar kostnaderna för att åtgärda dem.
4. **Fel hopar sig ofta:** Fel tenderar att samlas i vissa funktioner eller delar av systemet.
5. **Anpassa/variera test:** Teststrategier bör anpassas för att täcka olika scenarier och användningsfall.
6. **Test är kontextberoende:** Testmetoder och mål beror på typen av system och användningsområde.
7. **”Absence of errors” - fallacy:** Att inga fel hittas betyder inte att systemet uppfyller alla krav eller är användbart.



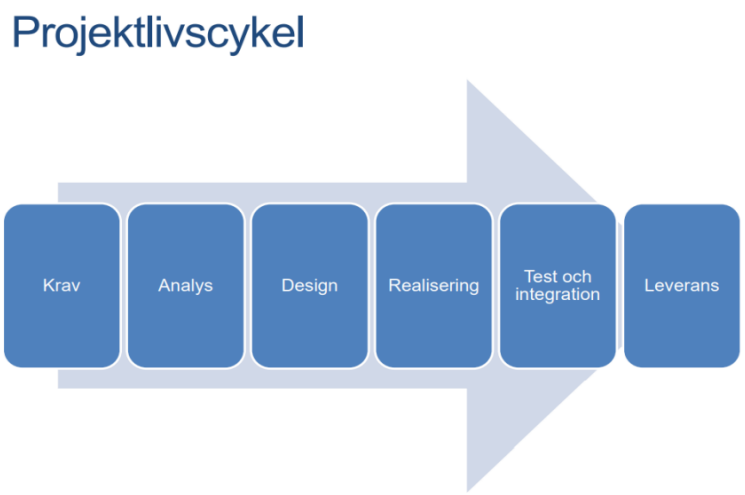
Konsekvenser av bristande test

- **Stora kostnader för utveckling:** Fler resurser krävs för att rätta till problem som hade kunnat upptäckas tidigare i processen.
- **Högre kostnader för förvaltning:** System som inte är ordentligt testade kräver mer underhåll och löpande korrigeringar, vilket ökar förvaltningskostnaderna.
- **Missnöjda kunder:** Brister i funktionalitet eller kvalitet kan leda till att kunder förlorar förtroendet för produkten eller tjänsten.
- **Missnöjda användare:** Problem med användbarhet, stabilitet eller funktioner påverkar användarnas upplevelse negativt och kan leda till minskad användning.

Var kommer (alla) buggar ifrån?

- **Tidspress:** När utveckling sker under tidspress kan tester förbises och slarviga lösningar implementeras.
- **Komplex kod:** Komplicerade kodstrukturer ökar risken för misstag och svårigheter att förstå och underhålla koden.
- **Komplex infrastruktur:** System med många beroenden, integrationer och distribuerade komponenter skapar fler möjligheter för fel att uppstå.

Projektlivscykeln



1. **Krav:**
   Samla in, definiera och dokumentera krav från intressenter.
   Skapa en kravspecifikation som fungerar som grund för resten av projektet.
2. **Analys:**
   Analysera kraven för att säkerställa att de är förstådda och genomförbara.
   Identifiera tekniska och affärsmässiga behov samt eventuella begränsningar.
3. **Design:**
   Utforma systemarkitektur, gränssnitt och funktionalitet baserat på krav och analys.
   Dokumentera designen som vägledning för utvecklingsfasen.
4. **Realisering:**
   Utveckla och implementera systemet enligt designen.
   Skriva och testa kod samt integrera olika komponenter.
5. **Test & integration:**
   Testa systemet för att säkerställa att det uppfyller kraven och fungerar korrekt.

Integrera delsystem och kontrollera att helheten fungerar smidigt.

6. **Leverans:**
   Överlämna den färdiga produkten till kunden eller användaren.
   Säkerställ att dokumentation och supportmaterial är tillgängligt.

Test vs Kvalitetssäkring

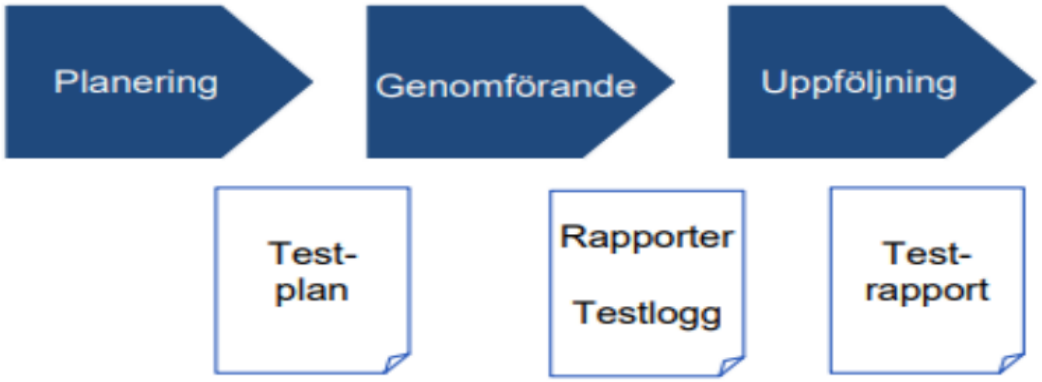
Test är en viktig del i kvalitetssäkringen, men det är inte hela processen. Kvalitetssäkring (QA) omfattar alla aktiviteter som syftar till att säkerställa att en produkt uppfyller de förväntade kvalitetskraven, och testning är en av dessa aktiviteter. *Testning syftar till att hitta fel i systemet medan kvalitetssäkring handlar om mer än bara testning och inkluderar bland annat kravhantering, projektledning och driftsättning.*

- För att uppnå god kvalitet krävs även välfungerande processer inom andra områden, såsom:
  - ***Kravhantering:*** *För att säkerställa att rätt krav är definierade från början.*
  - ***Projektledning:*** *För att hantera resurser, tid och risker under utvecklingsprocessen.*
  - ***Systemutveckling:*** *För att skapa robusta och hållbara systemlösningar.*
  - ***Driftsättning:*** *För att se till att systemet rullas ut och fungerar i produktionsmiljön.*
  - ***Användarhjälp och utbildning:*** *För att säkerställa att slutanvändare kan använda systemet effektivt och korrekt.*

Testprocessen & Testdokument

Planering – Genomförande – Uppföljning

Övergripande testprocess (forts)



(P)

Vad gör vi i planeringen?

- ***Definierar syftet med testet/testerna*** = Syfte (varför): vad ska det leda till…
- ***Identifierar testområden*** = t ex del av ett system. Kopplas till kravspec
- ***Granskar kravdokument*** = Kraven behöver vara ”testbara”
- ***Bedömer risker*** = Identifiera risker som kan påverka testarbetet; teknik, mm
- ***Skapar testunderlag*** = testfall, checklistor och testdata
- ***Säkrar resurser***

Alltså Planering = Övergripande planering → Skriver en testplan eller teststrategi. Detaljplanering → Testfall och checklistor. Skapa testmiljö → Skapa testmiljö och testa testmiljön

Testplan

En testplan är ett dokument som beskriver vad som ska genomföras,

Aktiviteter under testplaneringen (som punkterna ovan)

- Identifiera avgränsning, risker och mål för test
- Planera roller, ansvar och resurssäkring
- Schemalägg (scheduling)
- Fastställ mätetal, start/stopp kriterier, testnivåer
- Integrera testaktiviteter i utvecklingslivscykeln
- Identifiera risker och reservplaner
- Fastlägg uppföljningsmetodik

Syftet med testplanering är att skapa en vägledande plan för testgenomförandet. Den baseras på kravspecifikationer och andra dokument. Testplanen säkerställer att alla testaktiviteter täcker de nödvändiga funktionerna och kraven. Indatan är kravspecifikationer och design, och utdata är testplaner som kan delas upp beroende på projektets omfattning. Planen specificerar resurser, tidpunkter, testfall och förväntade resultat. Testplaneringen bör vara iterativ för bästa resultat.

Testplanens delar – kort beskrivning

- **Unik identifierare av testplanen:** Skapa logik för uppbyggnad av identifieraren
- **Inledning:** Beskriv bakgrund, syfte och mål samt hänvisning till andra dokument
- **Testobjekt:** Vad som ska testas, t ex system, komponent, etc (jfr förvaltningsobjekt)
- **Omfattning:** Vilka egenskaper som ska testas (t ex funktionella tester)
- **Avgränsning:** Vad ska inte testas
- **Tillvägagångssätt:** Arbetssätt; aktiviteter, tekniker, verktyg, mm. Beskriv hur urval av testfall o testdata gått till.
- **Start- och slutkriterier:** Bestäm kriterier
- **Avbrytande- och återupptagandekriterier:** Bestäm kriterier
- **Testdokumentation:** Vilka dokument som kommer att tas fram under testarbetet (artefakter / leverabler)
- **Testaktiviteter:** Detaljerade aktiviteter i de tre testfaserna.
- **Testmiljö:** Krav och utformning
- **Ansvar:** Roller och ansvar
- **Resurs- och utbildningsbehov:** Personer och kompetens
- **Tidplan:** Milstolpar och aktiviteter (varaktighet)
- **Risker och oförutsedda händelser:** Riskanalysen med åtgärder
- **Godkännande av testplanen:** Vem och när planen är godkänd.

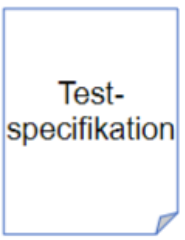
Varför planera? Skapa **förståelse** för den uppgift som ska lösas. Skapa **överblick** över det arbete som ska utföras. Skapa **underlag** för att avsätta resurser, fördela roller och ansvar i arbetet (organisera), kunna följa upp och ta konsekvenserna av förändrade förutsättningar

Alltid lika stor plan?

- *Ett kort projekt* med få deltagare behöver bara en översiktsplan
- *Ett långt projekt* med många deltagare behöver antagligen fler planer för att de skall vara hanterbara.

Testspecifikation

En testspecifikation är ett dokument som samlar alla *testfall* och beskriver detaljerna kring hur testerna ska utföras. Det fungerar som en sammanhållande



enhet för testfallen och ger en strukturerad överblick över testaktiviteterna(hittas längre ner). *En **testplan** fokuserar på **vad** som ska testas, som testtyper(integration, system & enhetstest), mål, resurser, tid och ansvar. **Hur** testerna utförs beskrivs i mer detaljerade dokument, som **testspecifikationer, testfall**.* Testspecifikationer innehåller typiskt följande delar:

1. **Inledning:**
   En kort beskrivning av testets syfte, omfattning och mål. Det kan också inkludera projektinformation och vem som ansvarar för testen.
2. **Testdata:**
   Specificering av den data som behövs för att köra testerna, inklusive testmiljöer och systemkonfigurationer. Testdata är avgörande för att kunna genomföra tester som representerar verkliga användarscenarier.
3. **Förberedelser:**
   Steg och åtgärder som behöver genomföras innan själva testningen kan påbörjas, till exempel installation av programvara eller konfigurering av testsystemet.
4. **Testfall:**
   Detaljerade beskrivningar av de tester som ska utföras, ofta baserade på kravspecifikationer eller användningsfall (UC). Varje testfall definierar testdata, steg för att utföra testet, förväntade resultat och kriterier för att bestämma om testet har passerat eller misslyckats. T.ex ett scenario där en viss funktion testas.

   - **Fördelar med testfall:**
     Strukturerat:
     Testfallen är tydligt definierade och organiserade, vilket gör testprocessen mer systematisk och lätt att följa.
     Bra grund för felrapporter:
     Eftersom testfallen är väl dokumenterade kan eventuella avvikelser enkelt identifieras och rapporteras med specifika detaljer om var och hur felet inträffade.
     Lätt att byta testare:
     En strukturerad testspecifikation gör det enkelt att överlämna tester till olika personer, eftersom alla steg och förväntade resultat är tydligt dokumenterade.
     Bra grund för automatisering:
     Testfall kan användas för att skapa automatiserade tester, vilket ökar testeffektiviteten och gör det lättare att återanvända tester för framtida versioner av programvaran.
   - **Nackdelar med testfall:**
     Blir snabbt omfattande:
     Om testfallet täcker många olika scenarier kan antalet testfall snabbt växa, vilket gör dokumentationen stor och svår att hantera.
     Kan bli svårt med överblick om testfallen är många:
     När det finns ett stort antal testfall kan det bli svårt att få en helhetsbild av teststatusen, vilket kan försvåra hanteringen och uppföljningen.
     Testfallen täcker inte allt – falsk säkerhet:
     Testfall är baserade på specifika krav eller scenarier, vilket innebär att de inte nödvändigtvis täcker alla möjliga användarscenarier eller systemfel, vilket kan ge en falsk känsla av att systemet är helt testat.
5. **Återställning**:

Efter att tester genomförts, ska systemet återställas till sitt ursprungliga tillstånd. Detta kan innebära att rensa testdata, återställa konfigurationer eller rulla tillbaka systemändringar för att säkerställa att testmiljön är redo för nästa testomgång.

(G)

Vad gör vi i genomförandet?

- Genomför test

Testgenomförande

Testgenomförande innebär ett systematiskt utförande av testfall, där testerna genomförs enligt den fastställda testplanen. Testerna utförs för att verifiera att systemet fungerar som förväntat, med kontroll av utfall och eventuellt skapande av avvikelse-/felrapport. Här är de viktigaste stegen i testgenomförandet:

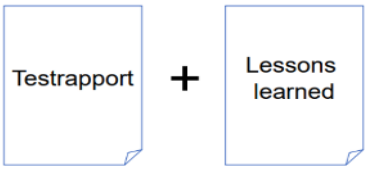
1. **Testa baserat på testfall:**
   Testerna genomförs genom att följa de **testfall** som definierats i testspecifikationen. Testfallen beskriver exakt vilka funktioner som ska testas, vad som ska göras och vad som förväntas som resultat.
2. **Använda planerade testdata:**
   Tester utförs med hjälp av de **testdata** som specificerats under planeringen. Testdata är avgörande för att simulera verkliga användarscenarier och för att kontrollera systemets funktionalitet i olika situationer.
3. **Manuella eller automatiserade tester:**
   Testerna kan genomföras **manuellt** av testare som interagerar med systemet, eller så kan de utföras med hjälp av **automatiserade tester**, där testprogram utför testerna utan mänsklig inblandning. Valet mellan manuella eller automatiserade tester beror på testets natur, komplexitet och behov.
4. **Fångande och rapportering av fel:**
   Under testningen **fångas fel** eller **avvikelser** som kan uppstå. När ett fel upptäcks, rapporteras det omedelbart, och detaljer om problemet (t.ex. steg för att reproducera felet, testdata, och eventuella felmeddelanden) dokumenteras noggrant.
5. **Felrapportering:**
   Fel rapporteras vanligtvis i en **dokumentmall** eller ett **felrapporteringsverktyg**. Dessa rapporter innehåller all relevant information om felet

Testlogg/Avvikelselogg = stor del av denna fas! - resultat av varje test loggas
Felrapport = även stor del.

(U)

Vad gör vi i uppföljningen?

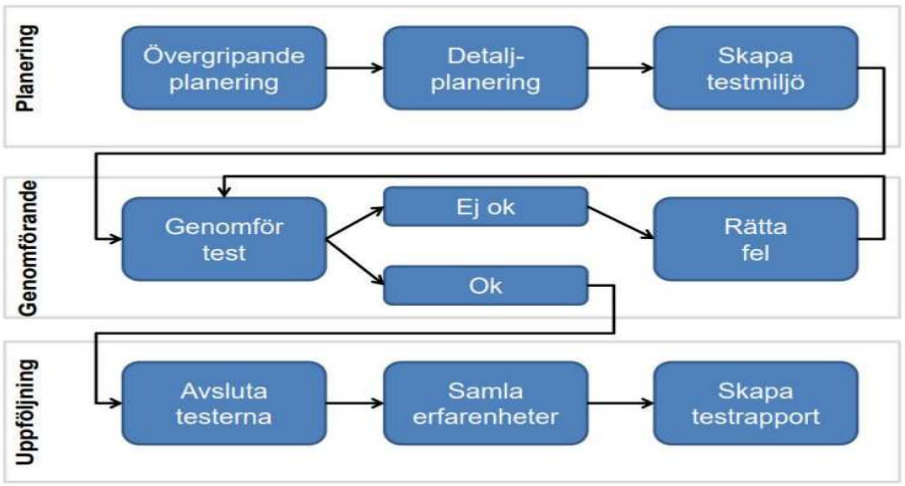
Sammanfattar: Sammanfattar testresultaten, inklusive vad som testades, resultat och upptäckta fel.
Utvärderar: Analyserar testprocessen för att bedöma effektiviteten och om målen uppnåddes.
Samlar erfarenheter: Dokumenterar lärdomar för att förbättra framtida tester och processer.



Avslutar test

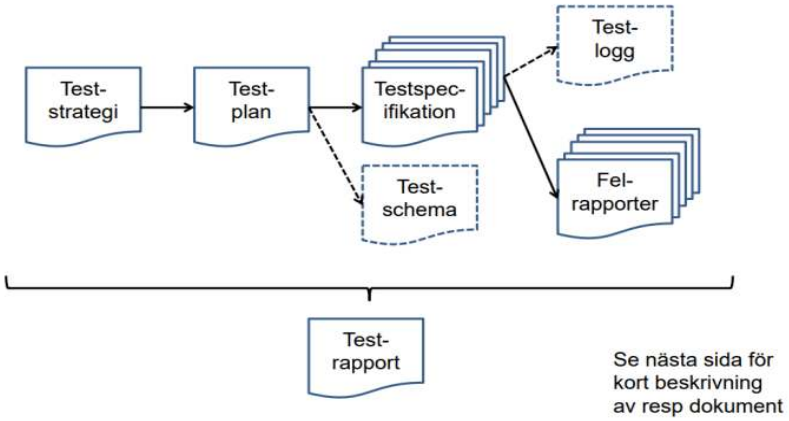
Testrapport = stor del i denna fas.

Testprocessen i mer detalj



Testdokument

Testdokument (alla delar)



Teststrategi — Generell och övergripande beskrivning om hur ett system vanligtvis testas → Testnivåer, roller och hur arbete brukar läggas upp.
Testplan — Beskriver vad som ska göras, av vem/vilka, när och varför
Testspecifikation — Innehåller testfall med testdata, förberedelser och återställning
Testschema — Beskriver i vilken ordning testfallen ska genomföras. (körschema)
Testlogg/Avvikelselogg — även kallat *testdagbok.* Här antecknas resultatet och händelser av varje testfall. Loggar alltså viktiga händelser under dagen, syftet är att testare ska kunna gå tillbaka och se vad som gjorts tidigare.
Felrapporter — En felrapport per hittat/identifierat fel

Testrapport — Sammanställning av resultatet av testerna.

1-4 (förberedelser i planeringsfasen)
5-6 (vad som sker under genomförandefasen)
7 (vad som skapas i uppföljningsfasen)

Testaktiviteter - en överblick

1. **Granska krav:**
   Gå igenom kravdokument för att förstå systemets funktionalitet och säkerställa att alla krav är testbara.
2. **Skapa testfall:**
   Utforma detaljerade testfall som täcker de olika funktionerna och scenarierna som ska testas.
3. **Prioritera testfall:**
   Bestäm vilka testfall som är viktigast att genomföra först, baserat på risk och vikt för systemet.
4. **Genomföra tester:**
   Utför testfallen enligt den planerade teststrategin och dokumentera resultaten.
5. **Logga resultat i testlogg:**
   Registrera resultaten från testerna, inklusive information om vilka tester som kördes, vilka som passerade eller misslyckades.
6. **Skriva avvikelserapport/felrapport:**
   Dokumentera eventuella problem eller buggar som upptäcks under testningen, med detaljer om hur de kan reproduceras.
7. **Kommunicera löpande med leverantören:**
   Håll kontinuerlig kontakt med utvecklingsteamet eller leverantören för att diskutera problem och lösa eventuella frågor som dyker upp under testningen.
8. **Sammanställa testlogg:**
   Sammanfatta all testinformation, inklusive resultat, avvikelser och status för testaktiviteterna.
9. **Skriva testrapport:**
   Skapa en slutrapport som sammanfattar testresultaten, inklusive vilka tester som genomfördes, eventuella problem som identifierades, och om systemet uppfyller kravspecifikationerna.
   - Unik identifierare
   - Inledning
   - Sammanfattning
   - Avvikelser
   - Bedömning av testernas omfattning
   - Sammanfattning av resultat
   - Utvärdering
   - Erfarenheter
   - Sammanfattning av aktiviteter
   - Godkännande

Problem med *ostrukturerad* testning

- **Trots tester upptäcks få fel:** Eftersom testningen är ostrukturerad kan det hända att många fel förblir oupptäckta, eftersom testerna inte täcker alla scenarier eller inte genomförs på rätt sätt.
- **Viktiga fel upptäcks sent:** Viktiga problem eller buggar kan upptäckas för sent i utvecklingsprocessen, vilket kan leda till kostsamma eller tidskrävande lösningar.
- **Test blir hinder för ”att bli klar”:** Ostrukturerad testning gör att det kan bli svårt att hålla testningen på schemat, vilket kan fördröja projektet och hindra leveransen.
- **Svårt att kontrollera och övervaka test:** Utan en tydlig plan eller struktur är det svårt att spåra teststatus, vilket gör det svårt att säkerställa att alla krav har testats ordentligt.
- **Lite användarmedverkan:** Om tester inte är välstrukturerade kan användare eller intressenter bli mindre involverade, vilket kan påverka kvaliteten på testerna och resultaten.
- **Testaktiviteter skjuts framåt i utvecklingsprojektet:** Ostrukturerad testning leder ofta till att testfasen försenas eftersom viktiga testaktiviteter skjuts fram, vilket kan påverka hela utvecklingsprocessen negativt.

Begrepp

Testdata

Testdata är den data som används för att genomföra tester på programvara.

- Behöver ha tillräckligt med testdata.
- Urval av testdata för aktuella tester
- Testdata bör vara ”produktionslik”.
- Testdata används för integrationstest, systemtest och acceptanstest. Även vid komponenttest
- Forma en process och/eller hantering för testdata, som är möjlig att använda och återställa.

Testmiljö

En testmiljö är en separat, isolerad miljö där tester kan genomföras utan att påverka den faktiska produktionsmiljön. Det är viktigt att testmiljön speglar produktionsmiljön för att resultaten ska vara tillförlitliga. Testmiljön används för att testa systemets funktioner innan de implementeras i produktion. Att ha en testmiljö gör det också möjligt att genomföra tester på nya funktioner eller uppdateringar utan att riskera systemets stabilitet i produktion. Deployment sker från utvecklingsmiljön till testmiljön och vidare till produktionsmiljön. (*skapas i planering- och används i genomförande fasen*)

Systemutvecklingsmetoder, Kravhantering, Test, och Granskning

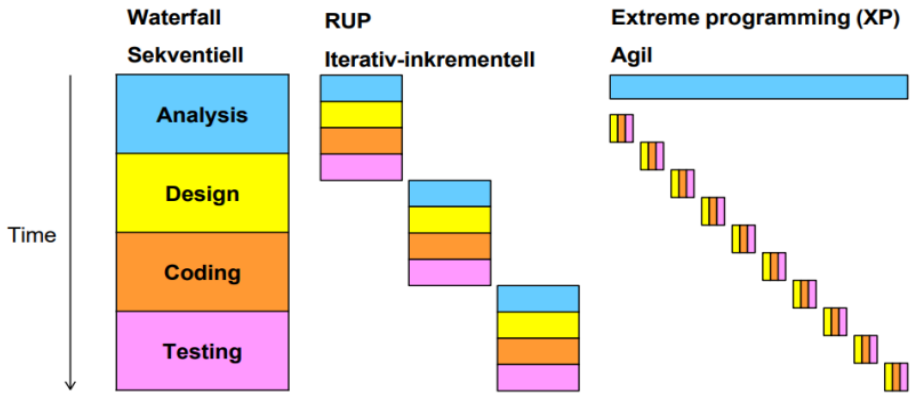
Systemutvecklingsmetoder - test

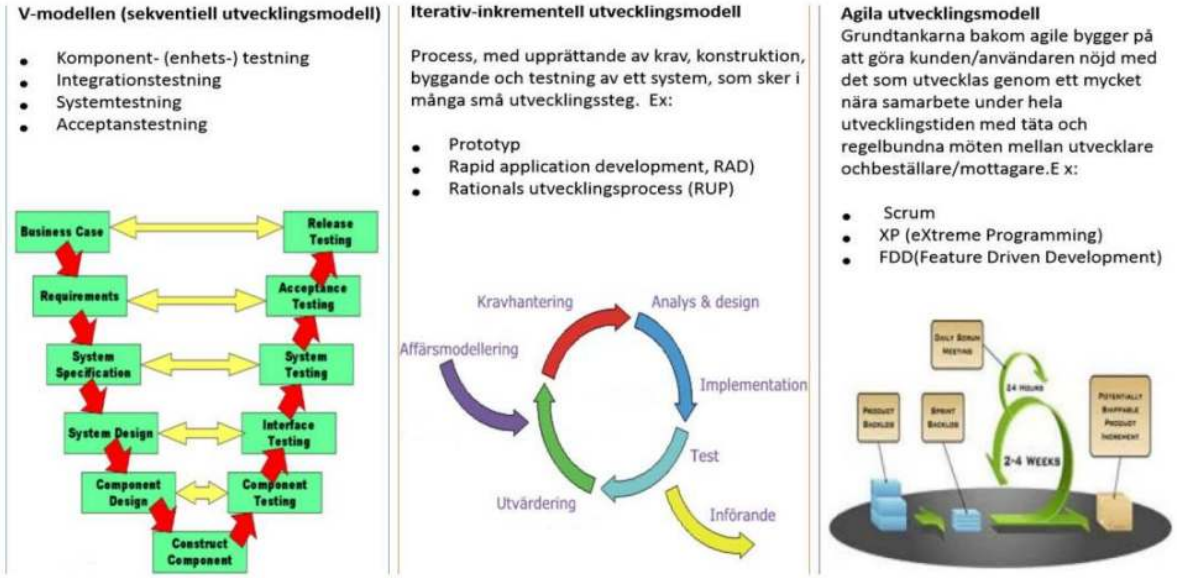
Systemutvecklingsmetoder och test

Det finns ett antal ”standardiserade” utvecklingsmetoder, där test utförs på lite olika sätt:

- Vattenfall
- RUP (Rational Unified Process)

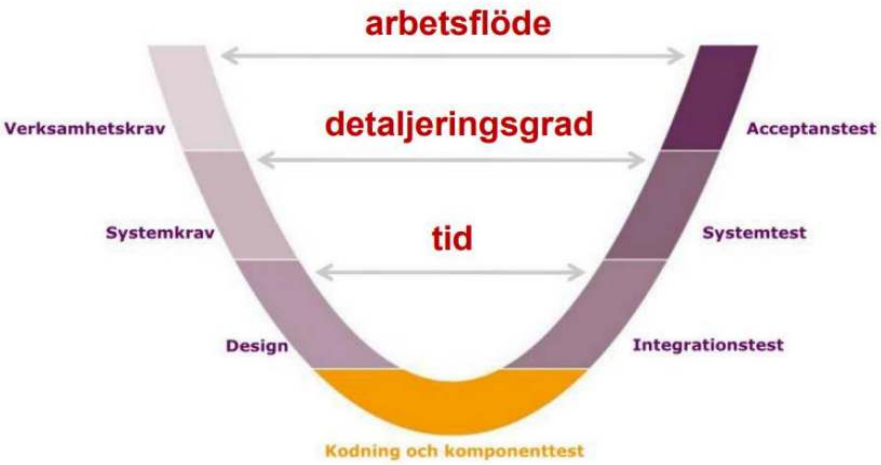
- XP (Extreme Programming)





V-modellen (Sekventiell)

- Linjär utvecklingsmetod (stegvis ordning där varje fas måste vara slutförd innan nästa påbörjas)
- Utveckling/Test enligt V-modellen.
- Test i slutet av utvecklingsperioden.
- Vattenfallsmodell.



Dimensioner i V-modellen

Arbetsflödet

Indelat i testnivåer; varje nivå har sitt syfte

Varje nivå kan ha testplan, testaktiviteter och testrapport

Detaljeringsgraden

Få detaljer på övergripande nivå

Detaljerat längst ned i modellen

Tiden

Test förbereds i den vänstra sidan av modellen

Test genomförs i den högra delen av modellen

Grundläggande testnivåer (testtyper)(ASIC)

Enhetstest eller Komponenttest — utförs av utvecklare för att testa individuella komponenter eller funktioner i systemet. Målet är att säkerställa att varje enhet fungerar korrekt isolerat, vilket gör att problem kan åtgärdas tidigt i utvecklingsprocessen.

Integrationstest (inom system) — fokuserar på att testa samverkan mellan olika komponenter i systemet. Det utförs ofta av utvecklare för att säkerställa att moduler fungerar tillsammans som de ska och att data flödar korrekt mellan dem.

Systemtest (mellan system, helheten) — genomförs för att testa hela systemet som en enhet. Testare verifierar att alla funktioner och krav är uppfyllda, och att systemet fungerar korrekt i en realistisk användarmiljö.

Acceptanstest — utförs av användare eller beställare för att säkerställa att systemet uppfyller användarnas behov och krav innan det går i produktion. Detta test bekräftar att systemet är redo för leverans.

Konsekvenser av vattenfallsmodellen – Blir bråttom i slutet. Test blir en regleringsfaktor beroende på tid som är krav. Om det hakar upp sig så tullar man på det som är i slutet. De som budgeterar för

utvecklingsprojekt älskar vattenfall. Detta eftersom man har kontroll på ekonomin. Man vet antal krav
och har en enklare mall på hur mycket det kommer kosta. Till skillnad för det agila där man har mindre koll på vad det faktiskt kommer att kosta. Man satsar på full kontroll i vattenfall. Vad är det vi satsar våra pengar på

RUP (Rational Unified Modelling Process) (Inkrementell/Iterativ)

RUP är en iterativ och flexibel mjukvaruutvecklingsmetod som delas in i fyra faser (IECT):

1. **Inception (Start):** Definiera projektets mål och krav.
2. **Elaboration (Utveckling):** Utveckla systemarkitektur och hantera risker.
3. **Construction (Byggande):** Implementera och testa systemet.
4. **Transition (Övergång):** Leverera systemet till användarna och sätt det i drift.

RUP är **anpassningsbar** och **iterativ**, vilket innebär att varje fas kan upprepas flera gånger för att förbättra systemet. Skapt för flexibilitet — En riskbaserad process.

Testdisciplinen inom RUP

fokuserar på att säkerställa kvalitet genom hela utvecklingscykeln:

- **Tidigt och kontinuerligt testande:** Tester genomförs redan under tidiga faser, som en del av byggandet av systemet, och upprepas vid varje iteration. Varje funktion testas efter utveckling.
- **Testdriven utveckling:** Tester skrivs baserat på krav och design innan kod utvecklas för att säkerställa att systemet uppfyller de definierade målen.
- **Testfaser:** Testning sker i alla faser av RUP: enhetstester under byggandet, integrationstester under elaboration, samt system- och acceptanstester under transition.

Detta tillvägagångssätt säkerställer att fel identifieras tidigt, och att systemet kontinuerligt testas för att möta kvalitetskrav.

XP (Extreme Programming) (Agil)

en **agil metod** som fokuserar på att **leverera värde snabbt** genom att arbeta i **korta sprinter**, vanligtvis på **ca. 4 veckor**. Målet är att skapa högkvalitativ mjukvara genom att kontinuerligt förbättra och justera arbetet baserat på feedback (kundinvolvering).

Nyckelprinciper

- **Fokus på kundens behov:** Leverera funktionalitet snabbt och ofta för att säkerställa att det som utvecklas verkligen ger värde för användaren.
- **Testning under hela utvecklingen:** Systemet testas kontinuerligt under varje sprint för att säkerställa att alla delar fungerar som de ska.
- **Parprogrammering och kodgranskning:** Två utvecklare arbetar tillsammans för att skriva kod, vilket förbättrar kodens kvalitet och minskar risken för fel. Betonar teamarbete, mod och respekt.
- **Korta iterationer:** Arbetet delas upp i små, hanterbara delar som utvecklas och testas i varje sprint.

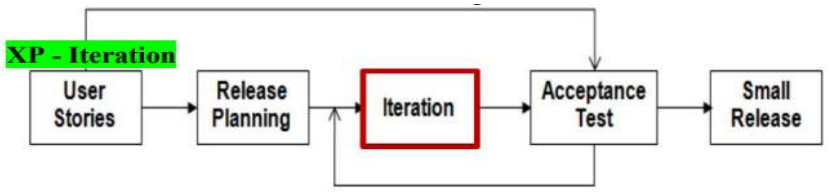
Extreme Programming (XP) förbättrar ett programutvecklingsprojekt genom att fokusera på fyra väsentliga dimensioner:

1. **Kommunikation:** XP uppmuntrar intensiv kommunikation mellan teammedlemmar, kunder och intressenter. Det innebär dagliga möten, nära samarbete mellan utvecklare och användare samt kontinuerlig feedback för att säkerställa att alla är på samma sida.
2. **Enkelhet:** XP främjar enkelhet i både design och kod. Målet är att skriva så lite kod som möjligt för att uppfylla aktuella krav, vilket minskar komplexitet och gör det lättare att underhålla och förändra systemet över tid.
3. **Feedback/återkoppling:** Tester och återkoppling är centrala inom XP. Testning sker kontinuerligt genom hela utvecklingsprocessen för att snabbt hitta och rätta till problem. Det hjälper till att säkerställa att programmet fungerar som förväntat och att förändringar kan göras snabbt.

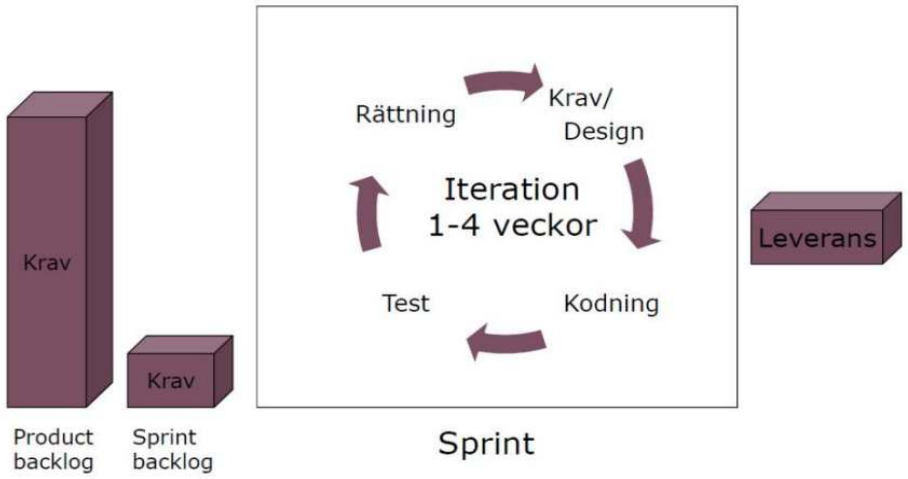
4. **Mod/respekt:** XP främjar mod i att göra förändringar när det behövs och att testa nya idéer utan att vara rädd för att misslyckas. Det innebär också respekt för teammedlemmarnas idéer och åsikter, samt att alla på teamet behandlar varandra med respekt för att skapa ett produktivt och kreativt arbetsklimat.

Testdisciplinen inom XP

Testdriven utveckling (TDD): Först skriver man test för att beskriva hur koden ska fungera, och sedan skriver man koden för att få testet att gå igenom.
Enhetstester: Man testar små delar av koden för att se till att allt fungerar som det ska.
Kontinuerlig integration: Man slår ihop och testar koden ofta för att säkerställa att systemet alltid är uppdaterat och fungerar.
Parprogrammering: Två utvecklare jobbar tillsammans och skriver både kod och tester, vilket gör att de kan hitta problem snabbt.
Regressionstester: Man testar att nya ändringar inte orsakar problem med funktioner som redan fungerar.



Agilt sätt att hantera krav & test



Test av IT-system - WWWWH

WHY (Syfte/n) — Syftet med testning av IT-system. Skäl/anledningar
WHO (Roller) — De som är involverade i testningen. Roller.
WHAT (Faser/Aktiviteter) — faser (planering, genomförande, uppföljning) och dess aktiviteter.
WHEN (Sekvens/Tidpunkt) — Testning sker vid olika tidpunkter i utvecklingscykeln: enhetstest utförs under utvecklingen, integrationstest när komponenter sätts ihop, systemtest när hela systemet är klart, och acceptanstest innan systemet tas i produktion.
HOW (Aktiviteter) — Testaktiviteter.

Roller inom test

Beställare: Ansvarar för att definiera krav och godkänna resultatet av tester, oftast genom acceptanstester.

Testare: Utför själva testningen av systemet, inklusive att köra tester och rapportera fel.
Testledare: Planerar, organiserar och övervakar testprocessen, säkerställer att tester genomförs enligt plan och att resurser finns tillgängliga.
Utvecklare: Skapar och testar enskilda komponenter av systemet (enhetstester och integrationstester).

Krav

Typiska problem och konsekvenser med system

Problem

- Insamlade kraven reflekterar inte det riktiga kundbehovet för nya systemet
- Krav är ofullständiga och/eller inkonsekventa
- Missförstånd mellan kunder och kravhanterare
- Addering av ej planerade förändringar till produktegenskaper – Om förändringar adderas utan att vara del av den ursprungliga planen kan det skapa förvirring, förlänga utvecklingen och öka kostnader.
- Oförmåga att spåra krav
- Varierande kravkällor – Om olika personer eller grupper bidrar med krav utan en tydlig samordning kan det leda till konflikt eller bristande överensstämmelse.

Konsekvenser

- Kunder och användare är inte nöjda med systemet – de använder inte alla systemfaciliteter eller de bestämmer att ersätta systemet med en annan lösning
- Systemet kan bli opålitligt, dvs. inkludera både logiska och tekniska problem
- Systemet kan bli levererat sent eller kan kosta mer än det var förhandlat

Olika typer av krav

Funktionella krav:

- Dessa krav beskriver **vad systemet ska göra**. De specificerar funktionerna som systemet måste stödja, såsom användarautentisering, databashantering, eller specifika affärslogikfunktioner. Exempel: "Systemet ska tillåta användare att logga in med användarnamn och lösenord."

Icke-funktionella krav:

- Dessa krav beskriver **hur systemet ska fungera** snarare än vad det ska göra. De inkluderar krav på prestanda, säkerhet, användbarhet, tillförlitlighet och skalbarhet. Exempel: "Systemet ska kunna hantera 1000 samtidiga användare utan att svarstiden överskrider 2 sekunder."

Normala krav:

- Dessa krav representerar de **grundläggande funktionerna och egenskaperna** som förväntas av systemet för att det ska kunna fungera korrekt under vanliga förhållanden. Krav som uttryckligen meddelats från beställare. Exempel: "Systemet ska kunna generera rapporter på begäran."

Förväntade krav:

- Dessa krav är de som **användare eller kunder förväntar sig**, men de kanske inte alltid är nödvändiga för systemets grundläggande funktionalitet. Alltså krav som inte uttryckligen meddelats från beställare men som ändå förväntas implementeras. De kan inkludera

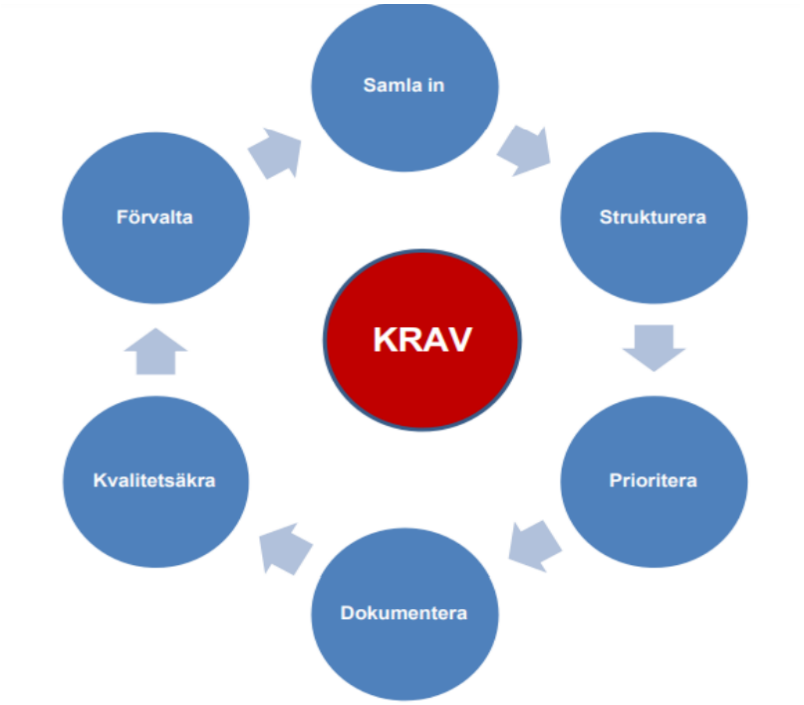
funktioner som förbättrar användarupplevelsen men som inte är avgörande. Exempel: "Systemet ska ha ett anpassningsbart användargränssnitt."

Sensationella krav:

- Dessa krav är **extra funktioner eller egenskaper** som inte förväntas men som kan ge systemet en unik fördel eller skapa en wow-effekt. Dessa krav kan vara "överdrivna" eller innovativa och är ofta designade för att skapa en stark positiv reaktion. Exempel: "Systemet ska kunna översätta alla användargränssnittet i realtid till valfri språk."

Kravstjärnan

Handlar om aktiviteter i samband med krav och kravinsamling



Sätt och tekniker för att samla in krav

SRS för gammalt system: Analys av kravspecifikationen (Software Requirements Specification) för det befintliga systemet för att identifiera vad som ska förbättras eller behållas.

Verksamhetsmodellering: Kartlägga verksamhetsprocesser och flöden för att identifiera krav som speglar affärsbehov och mål.

Intervjuer: Samtal med intressenter och användare för att få insikt om deras behov, förväntningar och problem med det nuvarande systemet.

Krav-workshop: Gruppmöten med intressenter för att diskutera och dokumentera krav gemensamt i en strukturerad process.

Prototyp-byggande: Skapa enkla versioner av systemet eller funktionerna för att få feedback och bättre förstå användarnas krav.

Scenarier/Storyboards: Visualisering av användarscenarier eller arbetsflöden för att tydliggöra hur systemet ska användas.

Brainstorming: Gruppdiskussioner för att generera idéer och krav på ett kreativt sätt utan initiala begränsningar.

Användningsfallsmodellering: Dokumentera användarnas interaktion med systemet i olika scenarier för att definiera funktionella krav.

Strukturera krav

Strukturering av krav är en aktivitet som pågår kontinuerligt. Aktiviteten går ut på att skapa struktur som är lätt att överblicka och förvalta. *Metoder*: Använd hierarkiska kategorier, grupperingar eller verktyg som kravhanteringssystem för att hålla krav lättillgängliga och spårbara.

Prioritera krav

Fokusera på det viktigaste, Hitta hög- och låg-prioriterade krav, Implementation av rätt krav i rätt ordning, Spara tid och pengar.

Dokumentera krav

Dokumentation säkerställer att alla krav är tydligt definierade, spårbara och begripliga för utveckling och test.

Former av dokumentation:

- **Kravspecifikation:** Formellt dokument med detaljerade krav.
- **Användningsfall:** Beskrivningar av hur användare interagerar med systemet.
- **Användarberättelser:** Enkla, korta berättelser om användarens behov och mål.

  Bör innehålla:
  - Funktionella och icke-funktionella krav.
  - Gränssnitt mot andra system.
  - Krav på användargränssnitt.
  - Designrestriktioner.

Utmaning:

- För lite dokumentation skapar **osäkerhet** om vad som ska levereras.
- För mycket dokumentation blir **svårhanterat** och gör det svårt att identifiera brister.

Kvalitetssäkra krav

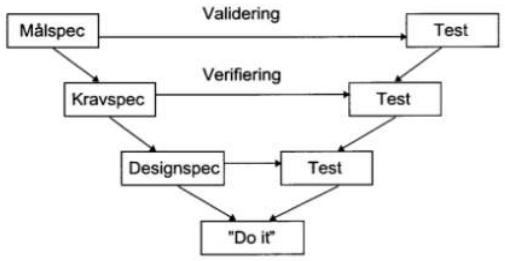
En ständigt fortlöpande aktivitet som syftar till att säkerställa att de dokumenterade kraven dels beskriver rätt egenskaper och dels beskriver dem på rätt sätt.

Validering – bygger vi rätt system?

- Uppfyller kravspecifikationen de verkliga behoven?
- Kommer beställaren att bli nöjd?
- Extern kvalitetssäkring

Verifiering – bygger vi systemet rätt?

- Uppfyller det färdiga systemet kravspecifikationen?
- Följer vi kravspecifikationen?
- Intern kvalitetskontroll – sker oftast inom utvecklingsteamet för att kontrollera att systemet byggs på rätt sätt enligt specificerade krav.



Validering
Gör vi rätt saker?
(kontrolleras mot kund)

Verifiering
Gör vi saker rätt?
(kontrolleras mot specifikation)

Det är viktigt med spårbarhet mellan krav och verifiering samt att alla krav testas på motsvarande nivå.

Förvalta krav – hantering av förändringar

Förvaltning av krav handlar om att säkerställa att förändringar hanteras strukturerat och effektivt under hela projektet.

1. **Konfigurationshantering:**
   - Skapa en tydlig översikt över aktuella krav så att alla inblandade har tillgång till uppdaterad information. Hålla koll på krav.
2. **Förändringshantering:**
   - Rutiner för att hantera ändringar i krav, inklusive godkännandeprocesser och dokumentation av ändringarna.
3. **Påverkansanalys:**
   - Utför en analys för att bedöma hur en förändring påverkar andra delar av systemet och projektet. Spårbarhetsmatris för att koppla krav till design, implementation och test, vilket underlättar att förstå effekterna av förändringar.

Förvalta krav – spårbarhetsmatris

Kopplar krav och testfall. En spårbarhetsmatris är ett verktyg som kopplar samman krav, design och tester för att säkerställa att allt hänger ihop. Den gör det möjligt att förstå konsekvenser av ändringar, till exempel om ett krav ändras kan det påverka testfall och leda till nya problem. Matrisen hjälper också till att säkerställa att alla krav har tillhörande testfall. Den kan skapas i Excel eller med hjälp av kravhanterings- och testverktyg.

	Testfall						
krav	1	2	3	4	5	6	7
1	X	X					
2	X	X	X	X	X	X	X
3			X				
4				X			
5					X	X	
6							X

Tekniken hjälper dig att fastställa ursprunget för varje krav. Samt att;

- Kontrollera att ett färdigt system uppfyller alla krav: - att allt som kunden begärt implementerats.
- Kontrollera att IT-systemet bara gör det som begärdes: att man inte implementerat något som kunden aldrig bett om.
- Hjälp vid förändringsledning: - när vissa krav förändras, vill vi veta vilka testfall som bör göras om för att testa denna förändring.

När är kravinsamlingen färdig?

En kravinsamling blir aldrig helt färdig - Det finns alltid flera åsikter - Nya möjliga funktioner - Det går aldrig att fullständigt specificera en produkt.

Tidig testdesign

1) Testare deltar i kravdiskussioner
2) Testare deltar när kraven skrivs
3) Testare deltar vid granskning av kraven
4) Testare planerar och skriver testerna tidigt, helst parallellt med kraven
5) Krav och test i samma verktyg

Test

”Framgångsfaktorer” för test

- Påbörja test tidigt
- Testa under hela utvecklingsprocessen - Hitta fel tidigt… - Begränsa kostnaderna för fel
- Struktur i testarbetet
- Arbeta metodiskt för att kvalitetssäkra systemet

Manuell testning vs Automatiserad testning

Manuell testning:

Manuell testning innebär att en QA-analytiker eller testare utför testerna för hand på programvaran för att hitta buggar och säkerställa att allt fungerar som det ska under utvecklingen.

Fördelar:

- **Användarcentrerad:** Testar användarupplevelse och interaktioner.
- **Flexibel:** Lätt att anpassa för nya eller ad hoc-tester.
- **Upptäcker oväntade fel:** Möjliggör kreativt felsökande och identifiering av problem som automatiserade tester kan missa.

Begränsningar:

- **Inte lätt att upprepa:** Manuell testning kräver ofta att samma tester utförs flera gånger, vilket kan vara tidskrävande och sårbart för mänskliga misstag.
- **Tidskrävande:** Eftersom tester måste göras för hand, tar de längre tid att genomföra.

- **Svårt att säkerställa isolering:** Tester kan vara svåra att isolera och säkerställa att de inte påverkas av externa faktorer.

Automatiserad testning:

I automatiserad mjukvarutestning skriver testare kod / testskript för att automatisera test utförandet. Testare använder lämpliga automatiseringsverktyg för att utveckla testskript och validera programvaran. Målet är att slutföra test utförandet på mindre tid

- **Fördelar:**
  - **Snabbare och mer effektiv:** Automatiserade tester kan köras snabbt och ofta, vilket sparar tid, särskilt när tester behöver upprepas.
  - **Finner buggar tidigare:** Eftersom tester kan köras kontinuerligt, upptäcks buggar snabbt.
  - **Högre kvalitet:** Automatiserad testning minskar risken för regressionsfel och gör det lättare att förbättra koden.
  - **Klarhet och dokumentation:** Automatiserade tester fungerar också som levande dokumentation för programvaran, vilket gör det lättare att förstå koden och hur den ska fungera.
  - **Möjlighet att refaktorera:** Automatisering gör det lättare att göra kodförbättringar utan att riskera att bryta funktioner.

Manuell testning är användbart för att testa användarupplevelse, flexibilitet och oväntade fel som är svåra att förutse med automatiserade tester. Det är också enklare för mindre projekt eller för tester som behöver anpassas snabbt.

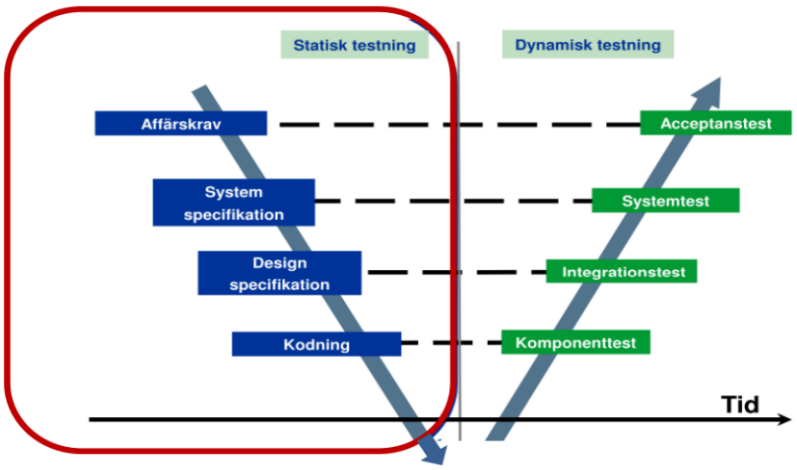
Statisk vs dynamisk testning

Statisk testning: Innebär att identifiera fel utan att köra programmet. Det handlar om att granska kod, design, och dokumentation för att hitta misstag och brister innan något körs. Detta kan göras genom kodgranskningar eller inspektioner där utvecklare eller testare går igenom materialet för att upptäcka problem tidigt.

Dynamisk testning: Exekvera programmet för att hitta fel. Innebär att faktiskt köra programmet för att se hur det fungerar i praktiken och hitta buggar genom att testa funktionaliteten. Här utförs tester som enhetstester, integrationstester och systemtester där programmet körs i olika miljöer och situationer för att säkerställa att det fungerar som det ska.

Static testing	Dynamic testing
Without executing the program	By executing the program
Analysis process	Verification process
Prevention of defects	Finding and fixing the defects
Evaluates code/documentation	Finds bugs/bottlenecks in the system
Cost of defect is less	Cost of defect is high
Return on investment will be high (early stage)	Return on investment will be low (after development phase)

Båda är viktiga!



Analogi: Bilkontroll = Att sparka på däcken, kontrollera färgen och titta under huven är en statisk testning. Att starta bilen, lyssna på motorn och köra på vägen är dynamisk testning.

Granskning

Granskning

Granskning genomförs för att säkerställa att dokument och kod är korrekta och möter kraven. Syftet är att verifiera att det som skapats är rätt, validera att det uppfyller verkliga behov, och skapa en samsyn bland projektmedlemmarna. Granskning är kostnadseffektivt eftersom det hjälper till att identifiera problem tidigt innan de blir dyra att åtgärda. Granskning kan göras på alla typer av dokument, som kravspecifikationer, design eller kod. Syftet är att hitta fel utan att behöva exekvera koden (*statisk testning, statisk analys*).

Målen med granskning: Verifiering, Validering, Konsensus (Nå samsyn), Förbättringar (Hitta förbättringsförslag), Felsökning (Hitta fel i dokumentation och kod innan felen byggs in i programmet).

När bör man granska?

Bör göras så tidigt som möjligt när det finns något meningsfullt att granska. Bör också göras ofta, eftersom det är lättare att granska små delar av dokument än stora. Regelbundna översyner ger snabb feedback till författarna.

Granskningstyper

Informell granskning: En enkel genomgång utan strukturerad process. Informell granskning används vid tidsbrist eller om dokumentet som granskas är mindre viktigt. - Informell granskning kan också användas som förgranskning till en mer formell granskning i syfte att beta av många småfel på kort tid (ex. stavfel och grammatiska fel). Billigaste sätt att uppnå viss nytta och att hitta fel.

Formell granskning: Mer organiserad och kan vara i form av en genomgång (walkthrough), teknisk granskning eller inspektion (mest formell). Formell granskning följer en dokumenterad granskningsprocess som exempelvis beskriver att det ska finnas möte, formella roller, krav på förberedelser samt mål. Ett exempel på en formell form av granskning är inspektion.

Genomgång (Walkthrough):

Författaren leder granskningen och förklarar sitt arbete, ofta utan tidsbegränsning. Syftet är

lärande, att hitta fel och skapa samsyn. Deltagarna är oftast oförberedda och deltar informellt.

Teknisk granskning:

Leds av en moderator och utförs av tekniska experter. Fokus ligger på att diskutera lösningar, hitta tekniska fel och dokumentera förslag till förbättringar. Kan vara både informell och formell.

Inspektion:

Den mest formella granskningen, ledd av en utbildad moderator. Följer en strikt process med förberedelser, checklistor och definierade roller. Målet är att hitta fel och följa upp med åtgärder.

Inspektion - fallgropar = Brist på utbildning, Bristande dokumentation, Brist på stöd från ledningen.

Granskningsprocess

1. **Planering =** I detta steg definieras syftet med granskningen, vilka dokument som ska granskas, och vilka deltagare som ska involveras. En plan skapas som inkluderar tidpunkter, ansvarsfördelning och kriterier för att avgöra om materialet är redo att granskas.
2. **Start =** Här introduceras granskningen för alla inblandade. Dokumenten distribueras till granskarna, och mötets mål, omfattning och regler för processen kommuniceras tydligt. Rollerna klargörs också.
3. **Individuella förberedelser =** Varje deltagare granskar materialet på egen hand, identifierar potentiella fel, oklarheter eller avvikelser från specifikationer. Observationerna dokumenteras och förbereds för att diskuteras under granskningsmötet.
4. **Granskningsmöte =** Deltagarna diskuterar de individuella observationerna. Fokus ligger på att identifiera och dokumentera fel, inte att lösa dem direkt.

   Deltagare:

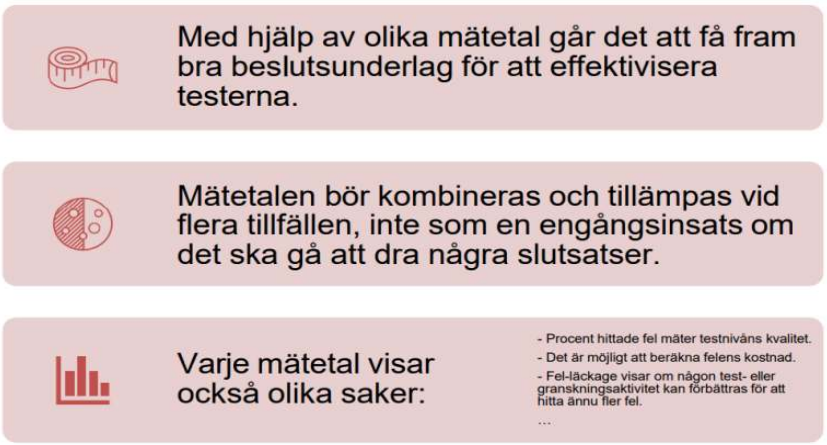
   - **Författaren**: Ansvarig för det granskade materialet, svarar på frågor men försvarar inte sitt arbete.
   - **Moderatorn (granskningsledare):** Leder mötet, ser till att processen följs och att diskussionerna är effektiva.
   - **Granskare**: De som analyserar materialet för att identifiera problem eller förbättringar (t.ex. testare, utvecklare eller kravanalytiker).
   - **Protokollförare (dokumentatör)**: Dokumenterar mötesdiskussioner, upptäckta fel och förbättringsförslag.
5. **Uppföljning =** Efter mötet följs eventuella åtgärder upp. Författaren av materialet reviderar det baserat på återkopplingen, och moderatorn kontrollerar att ändringar implementerats enligt rekommendationerna. Om det behövs kan en ny granskning planeras

Det som skiljer en inspektion från en genomgång

Det som skiljer en inspektion från en genomgång är att **inspektionen är en mer formell och strukturerad process.** Den har definierade **start- och slutkriterier**, och mötet leds av en **utbildad moderator** som inte är författaren av materialet. Vid inspektionen används **formella mötesroller**, **mätetal** för att bedöma effektiviteten och resultatet, och deltagarna genomför **noggranna förberedelser** innan mötet. Dessutom **protokollförs mötet** och alla **avvikelser** dokumenteras för uppföljning och åtgärder.

Mätetal

Mätetal (eller metrik) innebär att använda specifika, kvantitativa mått för att mäta och utvärdera ett system, en process eller ett resultat. I samband med granskningar och kvalitetssäkring kan mätetal användas för att objektivt bedöma effektiviteten, framstegen och kvaliteten på det som granskas. Exempel kan vara antal identifierade fel, tidsåtgång för att genomföra en granskning eller andel uppfyllda krav. Mätetalen hjälper till att strukturera och dokumentera granskningen för att kunna göra jämförelser och förbättringar över tid.



Framgångsfaktorer för granskning och granskningsmöten

Framgångsfaktorer för att lyckas med granskning och granskningsmöten inkluderar att genomföra grundläggande kontroller av dokumentet innan mötet och att ge rätt information till deltagarna i god tid. Det är också viktigt att välja rätt granskare och använda checklistor för att hålla processen strukturerad. För att undvika ineffektivitet bör långa diskussioner undvikas och mötet bör hållas inom fastställda start- och sluttider. Stöd från ledningen är avgörande för att säkerställa att granskningar genomförs effektivt, och det bör finnas en vilja att kontinuerligt förbättra granskningsprocessen.

Capability, Risk, Estimering

Capability

Värdering av processer (Capability)

Metoder för att mäta organisationers testmognad.

- ☐ CMM – Capability Maturity Model
- ☐ Inte nödvändigtvis bara för testprocesser utan för alla processer

CMM (Capability Maturity Model) (IMDQO)

beskriver fem mognadsnivåer för att förbättra och utveckla processer i en organisation:

1. **Initial (Mognadsnivå 1)** – Oplanerade och reaktiva processer. Arbete slutförs ofta, men är försenat och överskrider budget.
2. **Managed (Mognadsnivå 2)** – Processer hanteras på projektbasis. Projekten planeras, genomförs, mäts och kontrolleras.

3. **Defined (Mognadsnivå 3)** – Proaktiva processer. Standarder för hela organisationen ger vägledning för alla projekt och program.
4. **Quantitatively Managed (Mognadsnivå 4)** – Processer är mätbara och kontrollerade. Organisationen baserar sina beslut och förbättringar på kvantitativa data, vilket innebär att de använder mätningar och statistik för att övervaka och förbättra sina processer. Målet är att uppnå förutsägbara resultat som uppfyller både interna och externa intressenters behov.
5. **Optimizing (Mognadsnivå 5)** – Stabil och flexibel. På denna nivå är organisationen fokuserad på att ständigt förbättra sina processer. Den har en stabil grund men är också flexibel nog att anpassa sig till förändringar och nya möjligheter. Har en stabil grund för att vara agil och innovativ.

Risk

Risk och osäkerhet

Risk = sannolikheten att något negativt påverkar verksamhetens eller projektets möjlighet att nå sina mål - kan identifieras och hanteras.
Osäkerhet = man vet för lite om projektet för att kunna se riskerna.

Riskplanering/riskanalys (aktiviteter)

- Identifiera risker
- Analysera/värdera risker
    - Sannolikhet att risk inträffar
    - Konsekvens om risk inträffar
- Dokumentera resultatet
    - Risklista
- Planera för åtgärder

Riskbedöma

Enkel metod för riskhantering

- *Sätter ett värde mellan 1-5 på sannolikheten att risken inträffar*
- *Sätter ett värde mellan 1-5 på konsekvensen av att risken inträffar*
- *Riskvärdet är multipeln mellan sannolikhet och konsekvens (1-25, där 25 är värst)*

Riskutvärdering

Svara på frågor av typen;

- Vilken sannolikhet är det att risken inträffar?
- Vad får det för konsekvenser om risken inträffar? (kostnad, tid, ledtid, kvalitet)
- Vad kan vi göra för att undvika, möta, reducera eller acceptera risken?

Olika typer av risker

Tekniska: Risker kopplade till teknik, som komplexitet eller tekniska fel.
Projektorganisatoriska: Risker inom projektledning, som tidsplanering eller budgetöverskridande.
Linjeorganisatoriska: Risker i linjeorganisationen, som resurskonflikter eller bristande stöd.

Marknadsrelaterade: Risker kopplade till marknaden, som förändringar i efterfrågan eller konkurrens.
Kundrelaterade: Risker kopplade till kundens krav, engagemang eller förväntningar.
Externa: Risker från omvärlden, som lagändringar eller naturkatastrofer.
Projektrisker: Risker som påverkar projektets framgång, t.ex. förseningar eller fel leveranser.
Produktrisker: Risker kopplade till produkten, som kvalitet eller funktionalitet.
Humanrisker: Risker kopplade till individer, som brist på kompetens eller sjukfrånvaro.

Estimering

Estimera tid

För att säkerställa en effektiv planering behöver vi uppskatta hur lång tid varje aktivitet kommer att ta. Vissa aktiviteter kan kräva fler resurser, såsom extra personal, specifika lokaler eller verktyg. Dessutom är det viktigt att beräkna både tidsåtgång och kostnader för att kunna fördela resurser och hålla projektet inom budget.

- Estimering = UPPSKATTNING!!!
- Små aktiviteter är lättare att estimera än stora aktiviteter
- Enkla aktiviteter är lättare att estimera än komplexa aktiviteter
- Bryt alltså ner till enkla och korta aktiviteter!
- Utnyttja erfarenheter!

Olika sätt att estimera/uppskatta

- **Tidsskattning:** Gör en bedömning av hur lång tid varje aktivitet kommer att ta. Detta är grunden för att planera och allokera resurser.
- **Erfarenhetsbaserad uppskattning:** Använd erfarenheter från tidigare projekt för att förutse tidsåtgång. Erfarenheter kan hjälpa till att identifiera liknande aktiviteter och deras genomsnittliga tid.
- **Enpunkts-, tvåpunkts- och trepunktsskattning:**

  Enpunkts: En enda tidsuppskattning ges för en aktivitet. Enkel men mindre noggrann.
  Tvåpunkts: Man uppskattar både en optimistisk (snabbaste) och en pessimistisk (längsta) tidsåtgång.
  Trepunkts: Lägg till en "mest sannolik" uppskattning mellan optimistisk och pessimistisk för mer exakthet.
  - ***Optimistisk uppskattning (B):*** *Den kortaste möjliga tiden det kan ta, om allt går perfekt.*
  - ***Pessimistisk uppskattning (V):*** *Den längsta möjliga tiden, om saker går snett.*
  - ***Troligast (T):*** *Den tid som är mest realistisk baserat på erfarenhet och normala förhållanden.*
- **Uppräkningsfaktor:** Lägg till en marginal baserat på erfarenhet för osäkerhet.
- **Schablontid:** Använd standardiserade tidsramar från tidigare projekt.
- **Start- och slutkriterier kopplat till estimering:** När man uppskattar tid och resurser för en aktivitet är det viktigt att definiera start- och slutkriterier. Dessa kriterier hjälper till att klargöra när arbetet ska börja och vad som krävs för att det ska betraktas som färdigt. Genom att ha tydliga kriterier blir det enklare att beräkna den totala tidsåtgången.

Liechtenbergmetoden / Delphi-metoden (estimering)
Liechtenbergmetoden:

Den här metoden handlar om att använda erfarenhet och särskilja kända och okända faktorer. Den baseras på att uppskatta tidsåtgången genom att ta hänsyn till både bästa och sämsta scenarier.

- ***Optimistisk uppskattning (B):*** *Den kortaste möjliga tiden det kan ta, om allt går perfekt.*
- ***Pessimistisk uppskattning (V):*** *Den längsta möjliga tiden, om saker går snett.*
- ***Troligast (T):*** *Den tid som är mest realistisk baserat på erfarenhet och normala förhållanden.*

Denna metod ger en uppfattning om hur stor osäkerheten är genom att ta hänsyn till både de bästa och de värsta scenarierna, samt vad som är mest troligt.

Delphi-metoden:

Delphi-metoden bygger på att samla in expertbedömningar i flera omgångar för att nå en konsensus, vilket hjälper till att eliminera eventuella subjektiva eller biasfyllda uppskattningar.

- För varje aktivitet samlas optimistiska (B), pessimistiska (V), och troliga (T) uppskattningar från experterna.
- Genom att samla flera omgångar av bedömningar och bearbeta resultaten kan man gradvis minska osäkerheten och nå en mer exakt uppskattning av tidsåtgången.

Båda metoderna använder v,t,b uppskattningar, men Liechtenbergmetoden fokuserar på erfarenhetsbaserad uppskattning av risker och osäkerheter, medan Delphi-metoden fokuserar på att nå en enighet bland flera experter för att få en mer robust uppskattning.

Start & stopp/Avbrytande & återupptagande

- *Startkriterier* — Kriterier som ska vara uppfyllda för att inleda testerna (beroenden)
- *Stoppkriterier* — Kriterier som ska vara uppfyllda för att avsluta testerna
  - *Alla testfall är körda.*
  - *Inga kvarvarande öppna fel med högsta allvarlighetsgrad får förekomma*
  - *90% kodsatstäckning är uppnådd.*
- *Avbrytandekriterier* — Kriterier för att avsluta tester på ett onormalt sätt
- *Återupptagandekriterier* — Kriterier som ska vara uppfyllda för att återuppta testerna

---

Testtekniker och testverktyg

Testtekniker

Testdesigntekniker (upprepning)

Statisk testning: Innebär att identifiera fel utan att köra programmet. Det handlar om att granska kod, design, och dokumentation för att hitta misstag och brister innan något körs. Detta kan göras genom kodgranskningar eller inspektioner där utvecklare eller testare går igenom materialet för att upptäcka problem tidigt.

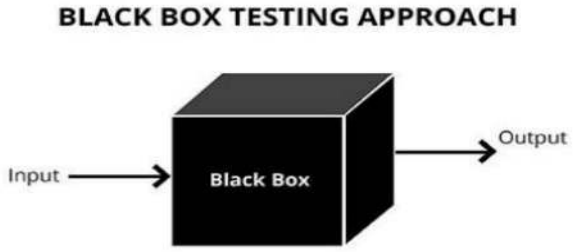
Dynamisk testning: Exekvera programmet för att hitta fel. Innebär att faktiskt köra programmet för att se hur det fungerar i praktiken och hitta buggar genom att testa funktionaliteten. Här utförs tester som enhetstester, integrationstester och systemtester där programmet körs i olika miljöer och situationer för att säkerställa att det fungerar som det ska.

Utforskande testteknik

Utforskande testning (Exploratory Testing) är en testmetod där testaren använder sin erfarenhet, kreativitet och intuition för att utforska applikationen utan att följa förutbestämda testfall. Under testningen lär sig testaren om systemets funktionalitet och identifierar problem genom att interagera med systemet.

- Frihet och ansvar att kontinuerligt förbättra sitt arbete genom att betrakta testrelaterad inlärning, testdesign, testutförande och tolkning av resultat som ömsesidigt stödjande aktiviteter, som pågår parallellt genom hela projektet.
- Under tiden ett program testas, så lär sig testaren nya saker som kan användas för att tillsammans med erfarenheter och kreativitet komma på nya, bra tester att utföra.
- Liknar Ad hoc – Latin för ”till detta” syftar på ”till detta speciella ändamål”. Ad hoc har fått dålig klang → Anses vara slarvigt…Ostrukturerat, tester utförs utan någon formell plan.

Black box-testning(1)
innebär att testaren inte har någon insyn i de interna strukturerna eller processerna i systemet. Istället baseras testningen på kravspecifikationerna. Syftet är att utvärdera funktionaliteten hos komponenten. Oavsett nivå i V-modellen kan man applicera black box testning. Testare är medvetna om vad komponenten ska utföra och arbetar emot variabler som ger förväntade resultat. Innebär att testaren vet vad systemet ska göra, men inte hur det gör det. *Black-box testning används ofta för systemtestning och acceptanstestning, där hela systemets funktionalitet testas från användarens perspektiv.*



Black box-testning (2) Här är några olika typer av testdesigntekniker som ingår i black box-testning:

- **Positiva tester:** Tester som verifierar att systemet fungerar korrekt när indata är korrekt och i enlighet med specifikationerna.
- **Negativa tester:** Syftar till att testa så att systemet kan hantera ett felaktigt beteende från användaren. Testar systemets felhantering. Testdata är felaktiga inmatningar eller felaktiga förutsättningar.
- **Extremtester:** Tester som använder extrema eller ovanliga värden för indata för att säkerställa att systemet klarar av ovanliga situationer utan att krascha eller ge oväntade resultat. *En variant av negativa tester.* Syftar till att försöka förstöra systemet.
- **Gränsvärdesanalys:** Tester som fokuserar på gränsvärden för indata, eftersom problem ofta uppstår vid dessa punkter (t.ex. minimum och maximum värden som systemet ska hantera).
- **Flödestester av webbaserade system:** Tester av hur användare interagerar med webbapplikationer, där man testar flöden som att navigera mellan sidor, fylla i formulär och skicka indata för att säkerställa att applikationen fungerar som förväntat.
- **Såpoperatester:** En teknik där man utför tester baserat på osannolika, udda eller orimliga scenarier för att se om systemet kan hantera situationer utanför normala användarscenarier.

  Såpopertest exempel ifrån kursboken:

  1) Boka en resa tur och retur mellan Stockholm och Örebro

2) Avboka resan.
3) Boka resan igen.
4) Ändra returresan till Gävle i stället för Stockholm.
5) Tåget blir inställt.
6) Boka om till ett senare tåg.

- **Säkerhetstester:** Tester som fokuserar på att identifiera säkerhetsbrister i systemet, som exempelvis otillåten åtkomst, dataintrång eller exponering av känslig information.

En nackdel med black-box metod är att det kan krävas ett oändligt antal testvärden för att täcka alla möjliga fel, vilket kan vara tidskrävande och svårt att genomföra.

White box-testning(1)(även känd som *Open Box*, *Clear Box*, eller *Glass Box*) är en typ av testning där testaren har full insyn i systemets interna struktur, kod och logik. Testaren fokuserar på att verifiera att varje enskild aspekt av komponenten fungerar korrekt. *White-box testning används ofta för enhetstestning och integrationstestning, där testaren fokuserar på att verifiera programmets interna funktionalitet.*

White box-testning(2)Här är några vanliga testdesigntekniker som ingår i white box-testning:

- **Kodtäckningsanalys:** Mäter hur mycket av koden som testas. Målet är att säkerställa att alla delar av koden (t.ex. funktioner, metoder, rader av kod) har testats för att identifiera potentiella brister eller obehöriga koder. Mäts i procent → högre desto bättre - Antalet kodsatser som körs dividerat med antalet kodsatser totalt i programmet. - En ”if-sats” räknas som en sats, oavsett vad som görs. - Om all kod körs minst än gång, blir kodsatstäckningen 100%.

  Cyklomatisk komplexitet är ett mått på koden som hjälper till att identifiera testbehovet. Ett mått på hur komplex en koddel är genom att räkna antalet möjliga vägar eller grenar som kan tas genom koden. Det är ett sätt att mäta mängden logik och beslutspunkter i en programkod, vilket hjälper till att bedöma hur många tester som behövs för att täcka alla möjliga vägar genom koden.

  - ↘ **Bra (1-7):** Om cyklomatisk komplexitet ligger inom detta intervall, anses koden vara enkel och lätt att testa.
  - ↘ **Fundera på att förenkla (8-10):** Här bör man överväga att göra koden enklare för att undvika för mycket komplexitet.
  - ↘ **Förenkla (>10):** Om komplexiteten överstiger 10 bör koden omstruktureras eller förenklas för att undvika svårigheter med testning och underhåll.
  - ↘ **Otestbart (>50):** Om komplexiteten är så hög som 50 eller mer, kan det vara mycket svårt eller nästan omöjligt att testa koden effektivt.
- **Kodgrenstestning:** Påminner om en kombination mellan kodtäckningsanalysen och Cycklomatisk komplexitet testning. Tester som syftar till att testa alla möjliga vägar eller grenar i koden (t.ex. if-else-satser, switch-case-strukturer) för att säkerställa att alla logiska vägar har testats.
- **Looptestning:** Tester som fokuserar på att säkerställa att alla loopar (t.ex. *for*, *while*) fungerar korrekt och att koden kan hantera olika iterationer utan att skapa oönskade resultat eller loopinfällningar.

  Enkla loopar: En loop som upprepar en viss uppsättning instruktioner flera gånger baserat på ett specifikt villkor.

  Nästlade loopar: Loopar där en loop är inuti en annan. Den yttre loopen körs först, och för varje iteration av den yttre loopen körs den inre loopen flera gånger.

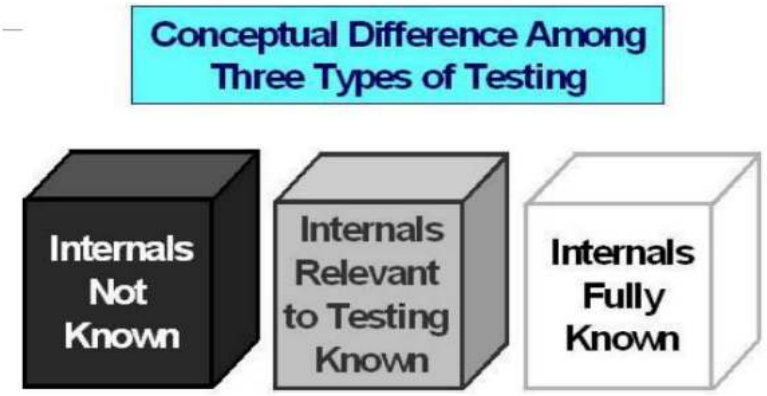
Konkatenerade loopar: Flera separata loopar som körs i följd utan att vara beroende av varandra. Varje loop körs oberoende av de andra.
Ostrukturerade loopar: Loopar med ett oklart eller svårt att följa kontrollflöde, vilket gör dem svåra att förstå och underhålla.

- **Villkorstestning:** Fokuserar på att testa alla villkor (t.ex. i if-satser eller while-loopar) för att verifiera att alla möjliga vägar genom koden har testats och att programmet fungerar som förväntat vid varje villkor.

En nackdel här är att det är svårt att upptäcka om krav saknas enbart genom att studera koden, eftersom denna metod inte alltid fokuserar på användarkrav eller systemkrav.

Gray box Kombinationen av **white box** och **black box** testning, även känd som **gray box** testning, anses ofta vara den bästa metoden. Den kombinerar fördelarna med båda tillvägagångssätten: Genom att kombinera dessa två metoder (black och white) får man en balans där man kan testa både funktionalitet och intern kodstruktur, vilket ger ett mer omfattande och effektivt testarbete.

Val av testtekniker
Komponenttest - White box, Black box
Integrationstest - White box, Black box
Systemtest - Black box



Faktorer som styr val av tekniker
Typ av system – Lagar, regler och standarder, Kundkrav eller kontrakt – Risker — Testdomän och syfte/mål med testerna — Tillgänglig dokumentation — Kunskap hos testarna — Tid och tillgänglig budget — Vald livscykelmodell — Struktur och kvalitet på kraven — Tidigare erfarenhet.

Testverktyg

Testverktyg programvara eller applikationer som används för att stödja och automatisera testprocessen i programvaruutveckling. De hjälper till att planera, genomföra och rapportera tester mer effektivt och noggrant.
Fördelar med testverktyg:

- Möjliggör tester som annars inte kan utföras manuellt.
- Möjliggör oftare och snabbare tester.
- Minskar repetitiva och tråkiga arbetsuppgifter.
- Ger utvecklare mer trygghet i testerna.

- Skapar objektiva bedömningar och resultat.

Nackdelar med testverktyg:

- Verktyg kan vara dyra att införskaffa.
- Kräver specialistkompetens för att sätta upp och underhålla automatiserade tester.
- Testskript måste kunna förvaltas för att inte bli föråldrade.
- Automatisering sparar inte alltid tid, särskilt vid initial uppsättning.
- Det finns en risk att förlita sig för mycket på automatiserade tester och missar viktiga manuella tester.
- Kan leda till att testtiden minskar, vilket gör att viktiga tester inte genomförs.
- Automatisering kan inte lösa problem om systemet har grundläggande kaos eller dålig struktur.

Översikt av testverktyg

- ***Kravhanteringsverktyg (kravdatabaser)*** = Underlättar överblick när kraven och personerna involverade i kravarbetet är många. I grunden en databas där man skriver in kraven. Hanterar ofta både krav, testfall och felrapporter →Möjliggöra spårbarhetsmatriser för statistik. - Hanterar konsekvenserna när krav ändras.
  Exempel*: RequisitePro(IBM Rational) ReQtest(Konsultbolag) Quality Center(HP Mercury) Doors(Telelogic) Visual Studio(Microsoft)*.
- ***Hantering av testfall =*** Skriver in testfallen i verktyget. Möjliggör en hierarki av testfallen, sortering och sökning. Ofta med stöd för dokumentgranskning. Håller reda på testfallets löpande status. Bör vara integrerat med hantering av felrapporter.
- ***Testdata =*** Verktyg för att fram testdata – Två huvudsakliga metoder;
  - Kopiering av data från driftmiljön till testmiljön - Görs oftast regelbundet med definierade rutiner för tillvägagångssätt
  - Användning av metadata som beskriver hur testdata skall se ut - Anger ett antal regler för hur data skall slumpas fram - Fördelar - Går snabbt - Nackdelar - Tar långt tid att definiera regelverket - Kan vara lämpligt vid nyutveckling
- ***Statisk analys =*** Används i syfte att underlätta granskning av programkod.Exempel på analys för viss typ av brister;
  - Avvikelser från utvecklingsstandard - Exempelvis namngivning av variabler.
  - Död kod - Kod som inte kan köras någonsin, då koppling för start saknas. Svårt att hitta manuellt
  - Oändliga slingor - Felaktiga villkor som aldrig kan inträffa.
  - Beräkning av komplexitet - Kan hitta kod som är mer komplex och därmed testa denna mer
- ***Dynamisk analys*** = genomförs när programmet körs för att identifiera brister som minnesläckage, stress- och prestandaproblem samt felsökning via debugger. Kodtäckningsanalys används för att mäta testtäckning genom att lägga till instrumenteringskod i systemet.
  Exempel på verktyg: Performance Monitor (Microsoft), Purify (IBM Rational)
- ***Testautomatiseringsverktyg =*** omvandlar manuella testfall till testprogram som både utför och kontrollerar tester. Dessa verktyg består ofta av flera delar, såsom inspelningsverktyg för att spela in testscenarier, redigeringsverktyg för att justera testfallen, uppspelningsfunktioner

för att köra testerna samt funktioner för att jämföra det verkliga resultatet med det förväntade resultatet.

- ***Felrapporteringsverktyg =*** gör det möjligt att följa ärenden från start till mål och minskar tiden för att felrapportera. De underlättar uppföljning och fördelning av arbetsuppgifter mellan testare, samt ger möjlighet att analysera trender för att ta informerade beslut under systemutvecklingsprocessen. Verktygen består ofta av delar som grundläggande arbetsflöde, formulär för felrapportering och funktioner för statistik, vilket också kan hjälpa till att svara på frågan om när tillräcklig testning har utförts.
- ***Testhanteringsverktyg =*** Eng. Test Management. Används för att styra och analysera testförloppet. Syftar till att underlätta skrivandet av testplan, testrapport och tidsplaner. Samlar även statistik från andra testverktyg om felrapportering, testfall och kravhantering. Ökar spårbarheten.
- ***Versionshanteringsverktyg =*** hanterar användning och ändringar av dokument av flera användare samtidigt, och bygger oftast på konceptet att checka in/ut dokument. Dessa verktyg används både för programkod och dokumenthantering. Exempel på versionshanteringsverktyg för programkod är Microsoft Team Foundation Server (TFS), Concurrent Versions System (CVS), Subversion (SVN) och GitHub. För dokumentfokuserade verktyg kan exempel vara Microsoft SharePoint, Dropbox och Google Drive.

Testverktyg (forts):
Komponenttest: Kodtäckningsverktyg, statiska analysverktyg.
Integrationstest: Testramverk, dynamiska analysverktyg.
Systemtest: Avvikelsehanteringsverktyg, testjämförare.
Acceptanstest: Testexekveringsverktyg, prestanda-/stressverktyg, säkerhetsverktyg
Verksamhetskrav/behov: Kravhanteringsverktyg
Systemkrav / krav och konstruktion: Testkonstruktionsverktyg, verifieringsverktyg.
Design: Testdataverktyg

Testfall och felrapport (det praktiska)

Testfall – Del av genomförande

Testfall
Vanliga fält

- ***ID*** = oftast ett nummer. Kan även använda prefix, ex. KU01 - Prefixet kan illustrera vilken systemdel som berörs.
- ***Rubrik*** = Viktigaste fältet. Skall vara beskrivande. Speciellt viktigt för att göra det möjligt att hitta ett visst testfall om det finns många testfall. Skall inte bara vara en förlängd form av ID. - Dåliga exempel: Testfall 1, 2 osv.
- ***Förberedelser*** = Aktiviteter som måste utföras innan testfallet kan utföras. Kan vara hänvisning till utförande av ett annat testfall -- Försök undvik för lång kedja, och för mycket upprepningar som istället kan samlas centralt i testspecifikationer.
- **Teststeg** = Instruktioner till testaren som skall utföra testfallet. Numrerad lista. Ca 3-8 steg i normalfallet - Exempel → Välj kundmodulen - Ange kunduppgifter - Klicka på knappen ”Spara”.

- **Förväntat resultat** = Används av testaren för att bedöma om testfallet har lyckats. Skall vara tydligt och konkret → Det skall vara lätt att avgöra om testets har lyckats eller inte. Lagom omfattande beskrivning.

  Dåligt förväntat resultat:
  "Systemet fungerar korrekt."
  Bra förväntat resultat:
  "Användaren loggas in och omdirigeras till startsidan, ett välkomstmeddelande med texten 'Välkommen, [Användarnamn]!' visas."
- **Återställning** = ”Motsats” till förberedelser. Instruktioner för att ”städa upp” efter utfört test. Så att inte andra testfall hindras från att kunna göras.
- **Testdata** = Kan ange specifika testvärden. Ännu bättre: en hänvisning till en datafil med testvärden →Ex. textfil, Excel-ark eller databas – Slippa redundanta testdata i testfall och underlätta förändringar. Syftar till att göra testerna återupprepningsbara
- **Prioritet** = Anger hur viktigt testfallet är → Bör baseras på prioriteringen i kravspecifikationen, Strategin för prioritering skall anges i testplanen. Kan användas för att ange ordning som testfallen skall köras. Viktigt om det föreligger tidsbrist
- **Kravreferens** = Refererar till vilket krav som testfallet berör - (Oraklet). Bör vara ett nummer. Kan användas för att skapa en spårbarhetsmatris.
- **Version** = Testfallen kan behöva skrivas om och förutsättningar ändras. Anger testfallets version. Är ett löpnummer. Viktigt för att testaren inte skall använda fel version av testfallet.
- **Författare** = För och efternamn på den som skrivit testfallet. Möjliggör att testaren kan fråga författaren direkt om vad som menas om testfallet uppfattas som otydligt.
- Typ av testfall =
- Konfiguration =

Typ av testfall Beskriver vad för sorts test som testfallet syftar till att göra, Ex. gränsvärde, stresstest, prestandatest osv. Underlättar strukturering när testfallen är många.

Alternativ till testfall Ad-hoc och checklistor.
Ad-hoc testning: Tester utförs utan förberedda testfall eller dokumentation. Det är mer informellt och används ofta för att snabbt hitta fel eller testa oplanerade scenarier. Svårt att återskapa fel och veta hur mycket av systemet som testats.
Checklistor: En mer strukturerad form av testning än ad-hoc. Mellanting mellan testfall och ad-hoc testning. Många korta testinstruktioner. Kan delas in i underrubriker för mer struktur. Bra kompromiss vid tidsbrist. Bättre än att fokusera på testfall för en del av systemet.

När används vilka testunderlag?
Komponenttester och integrationstester → Checklistor - (Ad-hoc tester)
Systemtester och acceptanstester (i fallande ordning) →Testfall (bäst) - Utforskande tester / Checklistor - Ad-hoc-tester (sämst)

(G) 2

Typiskt flöde testgenomförande

Tre huvudaktiviteter – Hur många iterationer körs. Förbättra programvara eller korrigera buggfixar.

Test – första test.

Omtest – Beroende på hur snabbt eller inte ett testfall lyckats eller kvarstår. Om fel kvarstår görs fler iterationer.

Regressionstest – Försöker köra funktionella och icke funktionella tester. För att se att det som utvecklats fortfarande funkar. Man måste se att det klaras av fortfarande efter ändringar.

Viktiga principer – genomförande

Omtest sker av; Helst den som rapporterat felet – Alt. Annan person på testsidan X Ej rättande utvecklare.

Arbeta i samma felrapporteringssystem; Ex. vid arbete med extern leverantör

Dagliga avstämningsmöten; Stämma av hinder, gårdagens och dagens arbete

Veckovisa redovisningar

Felrapport

När fel uppstår - Felrapport!

- **ID**: Ett unikt identifieringsnummer för felet.
- **Rubrik**: En kort sammanfattning av felet.
- **Felbeskrivning**: En detaljerad beskrivning av vad som är fel, inklusive steg för att återskapa problemet.
- **Systemdel**: Den del av systemet som felet påverkar.
- **Testobjekts version**: Den version av systemet där felet upptäcktes.
- **Bilagor**: Eventuella skärmdumpar, loggfiler eller andra dokument som stödjer felrapporten.
- **Testfalls-ID**: Koppling till testfallet som upptäckte felet.
- **Prioritet**: Hur brådskande det är att åtgärda felet (t.ex. låg, medel, hög).
- **Allvarlighetsgrad**: Hur allvarligt felet är för systemets funktion (t.ex. blockerande, kritiskt, mindre).
- **Åtgärdsbeskrivning**: Förslag på åtgärder för att fixa felet.
- **Namn på felhittare och ägare**: Den som upptäckte felet och den som är ansvarig för att åtgärda det.
- **Status**: Aktuell status för felet (t.ex. öppet, under utredning, stängt).
- **Datum öppnad/stängd**: När felet rapporterades och när det åtgärdades.

Exempel på felrapport

ID	14
Rubrik	Gick ej att spara kund med utländska tecken
Felbeskrivning	Teststeg: 1. Valde kundmodulen. 2. Klickade på knappen "Ny kund" 3. Fyllde i kundens namn och adress. 4. Klickade på knappen "Spara" Förväntat resultat: Att kunden sparades och att ett meddelande visades som sade att kunden blivit sparad. Verkligt resultat: Ett felmeddelande visades med texten "Ett fel inträffade vid försök att spara kund" Ytterligare info: Felet inträffar bara när kundens namn innehåller andra tecken än A-Z.
Systemdel	Kund
Öppnad av	Gunnar Gunnarson
Prioritet	Hög
Allvarlighetsgrad	1 – Kritiskt fel. Fel som orsakar att systemet eller en viktig komponent i systemet kraschar eller på annat sätt gör att systemet blir oanvändbart. Fel som leder till att data förstörs. Krasch, hängning eller förlorade data.
Prioritet	Hög – åtgärdas omedelbart.

Testtyper

Testnivåer - Vmodellen

Testnivåer/testtyper (forts) – bilden nedan viktig!

Ansvarsfördelning, normalfall.

	Samtliga utvecklare	En utsedd utvecklare	Testare	Användare
Komponenttest	X			
Integrationstest		X		
Systemtest			X	
Acceptanstest				X

Komponenttest(1) Kallas även →Enhetstest, Modultest, Programtest, Unit test

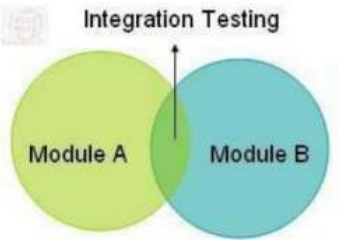
Komponenttest(2) Komponenttester fokuserar på att testa enskilda delar eller moduler av ett system, ofta genom att använda kod som är specifikt skriven för teständamål. Inleds så snart det finns några rader programkod. Dessa tester utförs vanligtvis med hjälp av testramverk som Nunit (för Visual Studio), Junit (för Java), MSTest, och andra, vilket möjliggör automatisering av testerna. Komponenttester kan omfatta flera olika typer av tester, till exempel:

- Test av enskilda funktioner (exempelvis att spara en kund eller ändra en sökning).
- Tester av enstaka sidor, som HTML-sidor eller skärmbilder.

- Tester av fält, exempelvis att kontrollera minimal och maximal längd på inmatning eller om fält för siffror accepterar bokstäver.
- Tester av klasser i objektorienterade system.

Dessa tester genomförs löpande av utvecklare och påbörjas så snart det finns kod att testa. Processen fungerar vanligtvis så att utvecklaren skriver kod och direkt därefter genomför tester, vilket kan innebära en cykel av att skriva kod och testa upprepade gånger. Målet är att snabbt identifiera och åtgärda problem i enskilda komponenter innan integrationstester påbörjas.

Tyvärr tenderar utvecklare ibland att prioritera att skriva mycket kod först och testa mer intensivt senare, vilket kan påverka testningens effektivitet.
Komponenttesterna avslutas när systemet går vidare till nästa steg, nämligen integrationstester.



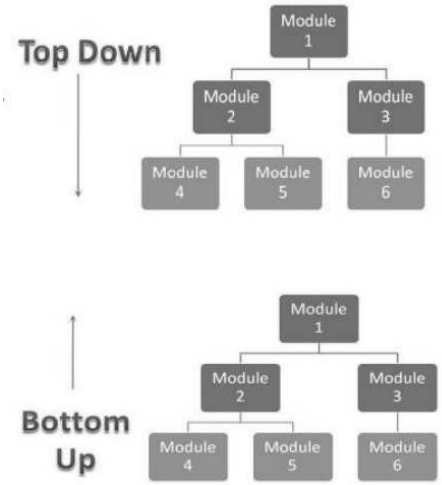
Integrationstest(1) Kallas även Komponentintegrationstest

- → Testar integration av komponenterna inom ett system Fel i komponenternas kommunikations-gränssnitt.
- → Utförs inte alltid, beroende på projektets storlek.

Integrationstest(2)

Här är en kort förklaring av de olika strategierna för integrationstestning:

1. **Big Bang**: Systemet byggs klart först, och sedan testas allt på en gång. Detta kan leda till svårigheter att identifiera var fel inträffar, eftersom all funktionalitet testas samtidigt.
2. **Stegvis integration**: En mer kontrollerad metod där systemet byggs och testas successivt. Det gör det enklare att hitta och isolera fel. Kan delas upp i:
   - **Bottom Up**: Testning börjar med de lägre nivåerna (t.ex. komponenter) och går uppåt. Testdrivrutiner används för att simulera de högre nivåerna.
   - **Top Down**: Testning börjar med de högre nivåerna (t.ex. användargränssnitt) och går neråt. Teststubbar används för att simulera de lägre nivåerna.
3. **Sandwich integration (Hybrid Integration Testing)**: En blandning av både nedifrån och upp samt uppifrån och ned, där man kombinerar fördelarna från båda strategierna för att hantera olika delar av systemet på ett effektivt sätt.



Systemtest(1) innebär att man testar hela systemet som en enhet för att säkerställa att det fungerar som förväntat i en komplett miljö. Det innefattar olika typer av tester:

- **Funktionella tester**: Testar om systemets funktioner och krav uppfylls enligt specifikationerna.
- **Icke-funktionella tester**: Fokuserar på systemets prestanda, säkerhet, användbarhet och andra kvaliteter som inte direkt rör funktionaliteten.
    - ***Prestandatest*** — Kommer systemet klara alla användare?
    - ***Stresstest*** —Hur systemet påverkas vid belastning som överskrider specificerade krav.
    - ***Användbarhetstest*** — Hur lätt eller svårt är det för användare att genomföra en viss uppgift?
    - ***Säkerhetstest*** — Klarar systemet av att hålla obehöriga ute?
    - ***Systemintegrationstest*** — Kopplingar till andra system.
- **Konfigurationstest**: Testar systemets funktion i olika miljöer och konfigurationer för att säkerställa att det fungerar korrekt under olika förhållanden.

Systemtest utförs oftast av ett testteam, snarare än utvecklarna själva, och syftar till att validera att alla delar av systemet fungerar tillsammans innan det går vidare till användartester eller produktionsmiljö.

Systemtest(3)

Fler icke funktionella testområden (*FURPS+ av Hewlett-Packard*)

Functionality - Capability (Size & Generality of Feature Set), Reusability (Compatibility, Interoperability, Portability), Security (Safety & Exploitability)

Usability (UX) - Human Factors, Aesthetics, Consistency, Documentation, Responsiveness

Reliability - Availability (Failure Frequency (Robustness/Durability/Resilience), Failure Extent & Time-Length (Recoverability/Survivability)), Predictability (Stability), Accuracy (Frequency/Severity of Error)

Performance - Speed, Efficiency, Resource Consumption (power, ram, cache, etc.), Throughput, Capacity, Scalability

Supportability - (Serviceability, Maintainability, Sustainability, Repair Speed) - Testability, Flexibility (Modifiability, Configurability, Adaptability, Extensibility, Modularity), Installability, Localizability

Acceptanstest(1) är en typ av test som vanligtvis utförs av användare för att säkerställa att systemet uppfyller deras krav och är redo att tas i bruk. Testet fokuserar på att validera systemet ur ett användarperspektiv och godkänna att det kan implementeras i den verkliga miljön. Testet utförs ofta genom scenarier som speglar verkliga användningssituationer och består vanligtvis av längre, sammanhängande aktiviteter snarare än enskilda enhetstester. Antalet scenarier är oftast begränsat, men de ska täcka alla viktiga funktioner och flöden för att säkerställa att systemet fungerar som förväntat.

Testtyper

Upprepning → Testtyper (förklarade i detalj på olika ställen i dokumentet):

Funktionella tester – Kontrollerar om systemet fungerar enligt kravspecifikationer och levererar rätt resultat.

Icke-funktionella tester – Utvärderar övergripande kvalitetsattribut som prestanda, säkerhet, användbarhet, tillförlitlighet och skalbarhet.

Kompontent/Unit/Enhetstester – Testar enskilda komponenter eller moduler för att säkerställa att de fungerar isolerat.

Integrationstester – Verifierar att flera komponenter fungerar korrekt tillsammans.
Systemtester – Testar det kompletta systemet som en helhet för att säkerställa att alla delar fungerar tillsammans som förväntat.
Acceptanstester – Utförs av slutanvändare eller kunder för att godkänna systemet innan det tas i bruk.
Regressionstester – Kontrollerar att nya ändringar inte har introducerat fel i tidigare fungerande delar av systemet.
Explorativa tester – Flexibel och kreativ testmetod där testare undersöker systemet utan fördefinierade testfall.

Mål för icke-funktionell testning (forts - upprepning??ta bort?)

- Bör öka produktens användbarhet, effektivitet, underhållbarhet och bärbarhet.
- Hjälper till att minska produktionsrisken och kostnaderna förknippade med icke-funktionella aspekter av produkten.
- Optimera hur produkten installeras, konfigureras, körs, hanteras och övervakas.

Icke-funktionell testning undersöker övergripande kvalitetsaspekter som inte direkt avser innehållet i systemets tjänster;

- **Användbarhet**: Hur enkelt och intuitivt systemet är för användarna att förstå och använda.
- **Robusthet**: Systemets förmåga att fortsätta fungera korrekt under olika förhållanden eller när det ställs inför oväntade situationer.
- **Tillförlitlighet**: Systemets förmåga att konsekvent utföra sina funktioner utan att krascha eller ge felaktiga resultat över tid.
- **Tillgänglighet**: Hur lätt det är att komma åt systemet, inklusive hur snabbt och pålitligt det är tillgängligt för användare under olika förhållanden.
- **Effektivitet**: Systemets förmåga att använda resurser (som tid och minne) på ett optimerat sätt utan att överbelasta eller slösa på resurser.
- **Flexibilitet**: Hur lätt systemet kan anpassas för att hantera förändrade krav eller nya situationer, såsom uppdateringar eller integrationer med andra system.
- **Skalbarhet**: Systemets förmåga att hantera ökad belastning eller växande mängder användare eller data utan att försämras i prestanda.
- **Säkerhet (safety)**: Systemets förmåga att förhindra oavsiktliga skador eller missbruk som kan påverka användarna eller systemets funktionalitet, såsom säkerhetsbrister vid drift.
- **Säkerhet (security)**: Skyddet mot obehörig åtkomst eller manipulation, inklusive skydd mot attacker och dataintrång för att säkerställa systemets integritet och konfidentialitet.

Målet med användbarhetstestning att tillfredsställa användarna och den fokuserar främst på följande parametrar för ett system:

- **Systemets effektivitet**
  - Är systemet lätt att lära sig?
  - Är systemet användbart och tillför värde för målgruppen?
  - Är innehåll, färger, ikoner och bilder estetiskt tilltalande?
- **Effektivitet**
  - Lite navigering bör krävas för att nå önskad skärm eller webbsida, och rullningsfält ska användas sällan.
  - Enhetlighet i formatet på skärmar/sidor i din applikation/webbplats.

    - Möjlighet att söka inom din programvara eller webbplats.
- **Noggrannhet**
    - Ingen föråldrad eller felaktig information, som kontaktinformation/adresser, bör finnas.
    - Inga trasiga länkar ska finnas.
- **Användarvänlighet**
    - Använda kontroller ska vara självförklarande och får inte kräva utbildning för att användas.
    - Hjälp ska tillhandahållas för användare att förstå applikationen/webbplatsen.
    - Alignment med ovanstående mål hjälper till med effektiv användbarhetstestning.

TDD & BDD

Test Driven Development (TDD)

Test Driven Development (TDD) / svenska = test driven utveckling

En metod där tester skrivs innan själva koden för att säkerställa funktionalitet och design. Här är huvudpunkterna:

1. **Ask questions early** – TDD tvingar utvecklaren att ställa viktiga frågor om design och krav tidigt i processen.
2. **Drives design** – Testerna styr systemets design och kodens struktur.
3. **Avoids accidental complexity** – Testdriven utveckling hjälper till att undvika onödig komplexitet genom att fokusera på de testade funktionerna.
4. **Fast feedback** – Snabb återkoppling gör det enkelt att åtgärda problem tidigt.
5. **Better code** – Koden blir mer genomtänkt och av högre kvalitet eftersom den alltid skrivs för att klara test.
6. **Complete regression suite** – TDD säkerställer att det finns en fullständig uppsättning tester för att undvika att nya förändringar bryter befintlig funktionalitet.

Sammanfattningsvis innebär TDD att skriva tester först för att säkerställa kvalitet och minimera buggar genom hela utvecklingsprocessen.

Användningsområde: Vanligt inom komponent- och enhetstestning.

Red-Green-Refactor

En central del av TDD och beskriver den cykliska process som används för att utveckla och förbättra koden.

1. **Red**: *Write a test that fails*. Först skriver du ett test som beskriver den funktion eller egenskap du vill implementera. Eftersom koden för den funktionaliteten ännu inte finns, kommer testet att misslyckas (det blir rött). Detta är helt förväntat, och det visar att ingen funktionalitet är implementerad ännu.
2. **Green**: *Make the code work*. Nästa steg är att skriva tillräckligt med kod för att få testet att passera. Här fokuserar du på att skriva den minsta mängd kod som krävs för att testet ska gå igenom. Du tänker inte på optimering eller förbättringar, utan bara på att få testet att bli grönt.
3. **Refactor**: *Eliminate redundancy*. När testet är grönt och har passerat, kan du börja refaktorera koden. Det innebär att du gör koden mer effektiv, läsbar eller underhållbar utan

att ändra funktionaliteten. Efter refaktoriseringen kör du testerna igen för att försäkra dig om att allt fortfarande fungerar som det ska.

Denna cykliska process upprepas varje gång du lägger till ny funktionalitet eller ändrar kod.

Behavior-Driven Development (BDD)

Behavior Driven Development (BDD)

En metod för test. I Behavior-Driven Development (BDD) skrivs scenarion först, och dessa används sedan som underlag för att skapa tester. Scenarion skrivs i ett strukturerat format som

Given-When-Then, vilket gör dem både lätta att förstå och använda som testunderlag.

Exempel:

```
Given en registrerad användare
When användaren anger rätt användarnamn och lösenord
Then ska användaren loggas in och visas sin startsida.
```

Koden som implementerar scenariona testas mot den utvecklade programvaran.

Version

Versionshantering

- innebär att tidigare versioner av dokument, källkodsfiler, program eller webbsidor kan återskapas, och ändringar gjorda i dessa tidigare versioner kan spåras.

Tekniska och politiska versioner

- ***Teknisk version***
  - ↷ varje gång man sparar
  - ↷ praktiskt men inget man fattar beslut kring
- ***Politisk version***

- när man ändrar nummer:
- Låser historiken
- Underlag för beslut
- Beskriver förändringar
- Kopplar ihop olika dokument
- Möjliggör ordnad förgrening

GAMLA TENTOR & KOMPLETTERING

Tenta 1

Uppgift 1 **(2 poäng)**

Förklara skillnaden mellan testfall och felrapport.

Testfall = innehåller detaljerade beskrivningar av de tester som ska utföras, ofta baserade på kravspecifikationer eller användningsfall (UC). Varje testfall definierar testdata, steg för att utföra testet, förväntade resultat och kriterier för att bestämma om testet har passerat eller misslyckats. T.ex ett scenario där en viss funktion testas.

Felrapport = Testfall är en grund för felrapport; Eftersom testfallen är väl dokumenterade kan eventuella avvikelser enkelt identifieras och rapporteras med specifika detaljer om var och hur felet inträffade. Felrapport dokumenterar eventuella problem eller buggar som upptäcks under testningen, med detaljer om hur de kan reproduceras.

Uppgift 2 **(2 poäng)**

Vilken roll är ansvarig för de olika typer av tester som beskrivs i V-modellen?

Komponenttest: Samtliga utvecklare.
Integrationstest: en utsedd utvecklare
Systemtest: testare
Acceptanstest: Användarna

Uppgift 3 **(2 poäng)**

I testarbetet utgör spårbarhetsmatrisen en central del. Förklara syftet med spårbarhetsmatrisen.

Syftet är att få en överblick och se att alla krav täcks av testfall. Gör det lättare att se vilka testfall som påverkas ifall ett krav ändras. Den gör det möjligt att förstå konsekvenser av ändringar, till exempel om ett krav ändras kan det påverka testfall och leda till nya problem. Matrisen hjälper också till att säkerställa att alla krav har tillhörande testfall.

Uppgift 4 **(2 poäng)**

Det finns olika roller som deltar i ett granskningsmöte. Ange två roller samt deras respektive uppgifter.

Granskning genomförs för att säkerställa att dokument och kod är korrekta och möter kraven. Syftet är att verifiera att det som skapats är rätt, validera att det uppfyller verkliga behov, och skapa en samsyn bland projektmedlemmarna (välj av dessa)

1. **Författaren**: Ansvarig för det granskade materialet, svarar på frågor men försvarar inte sitt arbete.
2. **Moderatorn (granskningsledare):** Leder mötet, ser till att processen följs och att diskussionerna är effektiva.
3. **Granskare**: De som analyserar materialet för att identifiera problem eller förbättringar (t.ex. testare, utvecklare eller kravanalytiker).
4. **Protokollförare (dokumentatör)**: Dokumenterar mötesdiskussioner, upptäckta fel och förbättringsförslag.
   (välj 2 av dessa om det kmr på tentan)

Uppgift 5 **(2 poäng)**

Vad är huvudsyftet med test av programvara och IT-system?

- För att identifiera fel.
- För att säkerställa kvalitet.
- För att skapa ett användbart system.

Uppgift 6 **(2 poäng)**

Vad innebär det att identifiera testområden i samband med förberedelser inför test?

- Att identifiera testområden innebär att **definiera vilka delar av systemet, funktionaliteten eller processen som ska testas**. Samt att **Prioritera områden** – riskområden som kräver extra fokus, exempelvis där det finns nya funktioner eller tidigare problem. **Matcha mot krav** – Säkerställa att alla definierade krav, både funktionella och icke-funktionella, täcks av testning. **Identifiera testtyper** – Bestämma vilken typ av test som behövs för varje område, exempelvis funktionstest, prestandatest eller säkerhetstest.

Uppgift 7 **(3 poäng)**

Beskriv med egna ord hur ett testgenomförande brukar se ut.

Testgenomförande innebär ett systematiskt utförande av testfall, där testerna genomförs enligt den fastställda testplanen. Testerna utförs för att verifiera att systemet fungerar som förväntat, med kontroll av utfall och eventuellt skapande av avvikelse-/felrapport. Här är de viktigaste stegen i testgenomförandet:

1. **Utföra tester enligt testplanen**: Genomför testfall enligt den fastställda testplanen för att verifiera systemets funktionalitet.
2. **Kontrollera resultat**: Jämför de faktiska resultaten med förväntade resultat för att säkerställa att systemet fungerar korrekt och uppfyller krav.
3. **Fånga fel och avvikelser**: Identifiera och dokumentera fel eller avvikelser som uppstår under testningen.
4. **Rapportera fel**: Skapa detaljerade felrapporter med information om hur felet uppstod, testdata och reproduktionssteg.
5. Testlogg!

Uppgift 8 **(3 poäng)**

Ange tre företeelser som skiljer en genomgång från en inspektion.

Genomgång (Walkthrough): Författaren leder granskningen och förklarar sitt arbete, ofta utan tidsbegränsning. Syftet är lärande, att hitta fel och skapa samsyn. Deltagarna är oftast oförberedda och deltar informellt.
Inspektion: Den mest formella granskningen, ledd av en utbildad moderator(ej författaren). Följer en strikt process med förberedelser, checklistor och definierade roller. Målet är att hitta fel och följa upp med åtgärder.

Uppgift 9 **(3 poäng)**

Om förutsättningarna förändras under ett testprojekt behöver testplanen revideras. Vilka huvudalternativ finns det när testplanen behöver revideras under ett testprojekt?

Uppdatera testomfånget Om projektkraven eller systemets funktionalitet förändras kan testomfånget behöva justeras. Detta innebär att lägga till nya testfall, ta bort eller ändra befintliga testfall för att säkerställa att alla förändringar täcks.
Justera testtidsramar Om tidslinjen för projektet ändras (t.ex. vid fördröjningar eller accelererade deadlines) kan testplanen behöva justeras för att anpassa testaktiviteterna till de nya tidsramarna. Detta kan innebära att omprioritera tester eller förändra testens omfattning.
Ändra testresurser Om det sker förändringar i tillgängliga resurser (t.ex. testteamets storlek, kompetens eller verktyg) kan testplanen behöva uppdateras för att reflektera dessa förändringar. Det kan innebära att omfördela uppgifter eller anpassa användningen av automatiserade tester.
Anpassa teststrategi eller metodik Om projektets riktning eller utvecklingsmetod (t.ex. från vattenfallsmodell till agil utveckling) förändras kan testplanens strategi behöva revideras för att passa den nya metodiken.

Uppgift 10 **(3 poäng)**

När testerna är genomförda skrivs en sammanfattande testrapport.
Ange tre uppgifter som måste dokumenteras och förklara dem kortfattat.

Sammanfattar: Sammanfattar testresultaten, inklusive vad som testades, resultat och upptäckta fel.
Utvärderar: Analyserar testprocessen för att bedöma effektiviteten och om målen uppnåddes.
Samlar erfarenheter: Dokumenterar lärdomar för att förbättra framtida tester och processer.

Uppgift 11 **(3 poäng)**

Ange tre huvudsakliga problem med ostrukturerad test och beskriv vad respektive problem får för konsekvenser.

Trots tester upptäcks få fel: Eftersom testningen är ostrukturerad kan det hända att många fel förblir oupptäckta, eftersom testerna inte täcker alla scenarier eller inte genomförs på rätt sätt.

Avgörande/viktiga fel upptäcks sent: Viktiga problem eller buggar kan upptäckas för sent i utvecklingsprocessen, vilket kan leda till kostsamma eller tidskrävande lösningar.
Test blir hinder för ”att bli klar”: Ostrukturerad testning gör att det kan bli svårt att hålla testningen på schemat, vilket kan fördröja projektet och hindra leveransen.
Svårt att kontrollera och övervaka test: Utan en tydlig plan eller struktur är det svårt att spåra teststatus, vilket gör det svårt att säkerställa att alla krav har testats ordentligt.
Lite användarmedverkan: Om tester inte är välstrukturerade kan användare eller intressenter bli mindre involverade, vilket kan påverka kvaliteten på testerna och resultaten.
Testaktiviteter skjuts framåt i utvecklingsprojektet: Ostrukturerad testning leder ofta till att testfasen försenas eftersom viktiga testaktiviteter skjuts fram, vilket kan påverka hela utvecklingsprocessen negativt.
(välj 2 av dessa om det kmr på tentan)

Uppgift 12 **(4 poäng)**

Identifiera och beskriv de olika strategierna för integration av olika komponenter inom ett system.

Big Bang: Systemet byggs klart först, och sedan testas allt på en gång. Detta kan leda till svårigheter att identifiera var fel inträffar, eftersom all funktionalitet testas samtidigt.
Stegvis integration: En mer kontrollerad metod där systemet byggs och testas successivt. Det gör det enklare att hitta och isolera fel.
Sandwich integration (Hybrid Integration Testing): En blandning av både nedifrån och upp samt uppifrån och ned, där man kombinerar fördelarna från båda strategierna för att hantera olika delar av systemet på ett effektivt sätt.

Uppgift 13 **(6 poäng)**

Ge exempel på hur användningen av olika verktyg kan underlätta de följande testnivåerna: a) komponenttest, b) integrationstest, c) systemtest, d) acceptanstest.

a) **Komponenttest** = **Kodtäckningsverktyg**: Mäter hur mycket av koden som testas och säkerställer att alla delar testas. **Statisk analysverktyg**: Identifierar kodproblem innan körning, som potentiella buggar eller säkerhetsrisker.
b) **Integrationstest** = **Testramverk**: Standardiserar och effektiviserar testskrivning och körning. **Dynamiska analysverktyg**: Upptäcker problem under körning, som minnesläckage eller felaktig interaktion.
c) **Systemtest** = **Avvikelsehanteringsverktyg**: Spårar och hanterar fel och avvikelser effektivt. **Testjämförare**: Jämför faktiska och förväntade resultat för att snabbt identifiera fel.
d) **Acceptanstest** = **Testexekveringsverktyg**: Automatiserar körning av testfall för snabb och upprepad verifiering. **Prestanda-/stressverktyg**: Testar systemets prestanda under belastning för att hitta flaskhalsar. **Säkerhetsverktyg**: Identifierar säkerhetsbrister och säkerställer att systemet är skyddat mot attacker.

Uppgift 14 **(6 poäng)**

På vilka sätt skiljer sig testarbetet mellan ett omoget och ett moget system? Ange tre sätt per typ av system.

Moget system – ett stabilt och väletablerat system som har genomgått flera förbättringscykler och är väl dokumenterat. De flesta större fel är åtgärdade och systemet har en beprövad funktionalitet.

1. **Stabilitet och dokumentation**: Testningen är mer systematisk och baseras på etablerade testfall och dokumentation.
2. **Reducerad frekvens av stora förändringar**: Färre större förändringar, testfokus ligger på regression och verifiering av nya funktioner.
3. **Färre kritiska buggar**: Testning handlar mest om att säkerställa att nya funktioner fungerar och att äldre funktionalitet inte är påverkad

Omoget system – ett system i tidig utvecklingsfas, ofta instabilt och med många funktioner under utveckling. Det saknar ofta fullständig dokumentation och har flera kända fel. Förändringar sker ofta för att förbättra systemets funktionalitet

1. **Högre grad av förändringar**: Systemet förändras ofta, vilket innebär att tester måste uppdateras kontinuerligt för att spegla dessa förändringar.
2. **Ofta många fel och brister**: Tester fokuserar på att identifiera och åtgärda buggar, eftersom omogna system ofta innehåller många fel.
3. **Bristande dokumentation och teststrategi**: Teststrategier och dokumentation saknas ofta, vilket gör att tester måste utvecklas parallellt med systemets utveckling.

Uppgift 16 **(9 poäng)**

För att förstå testning i sitt sammanhang är det viktigt att se testning som en integrerad del i systemutvecklingsarbetet. Den så kallade V-modellen är ett sätt att illustrera hur kravhantering och test hänger ihop med systemutveckling.
V-modellen har sju eller nio faser beroende på tolkning. Ange och beskriv kortfattat faserna (sju eller nio faser du väljer själv).

Verksamhetskrav = Insamling, prioritering, dokumentation och granskning av krav från beställare. Resultatet här är ofta en kravspecifikation eller ett antal användningsfall. Denna fas delar nivå med Acceptanstest där dessa krav testas.
Systemkrav = här samlas kraven in på systemets övergripande funktioner. Denna fas delar nivå med systemtest.
Design = på en mer detaljerad nivå beskrivs det hur systemet ska fungera. Hör ihop med integrationstest.
Kodning och komponenttest = i denna fas sker kodning och utvecklandet av systemet. Samtidigt sker komponenttest av utvecklarna för att testa varje enskild komponent för sig.
Integrationstest = här testar man hur komponenterna inom systemet fungerar ihop och samarbetar.
Systemtest = här testas systemet som helhet och dess funktioner, samt hur systemet fungerar med andra system.
Acceptanstest = användarna testar så att systemet gör det de ska och lever upp till kraven på funktioner och användningsfall.

Tenta 2

Uppgift 2 **(2 poäng)**

Nämn två alternativa benämningar på testnivån komponenttest.

Enhetstest, Modultest, Programtest, Unit test
(välj 2 av dessa om det kmr på tentan)

Uppgift 3 **(2 poäng)**

Nämn en variant av negativa tester och beskriv dess syfte.

Extremtester: *En variant av negativa tester.* Tester som använder extrema eller ovanliga värden för indata för att säkerställa att systemet klarar av ovanliga situationer utan att krascha eller ge oväntade resultat. Syftar till att försöka förstöra systemet

Uppgift 5 **(2 poäng)**

Vad menar konsultbolag1 med att man bör bli mer ”tvåbent” inom kravhantering & test?

Det är fördelaktigt att ha kunskap inom både kravhantering och testhantering. Om kravanalytikern har kunskap om testning kan denne skriva kraven på ett sådant sätt att testaren enkelt förstår funktionen som ska testas. Medan en testare som har kunskap om krav enkelt kan förstå sig på vad kravet syftar på.

Uppgift 6 **(2 poäng)**

Varför kan inte statisk analys hitta alla programvarufel?

Man kan inte förutse alla fel genom att endast granska koden. Man måste även köra programmet för att se hur den hanterar input. Statisk analys blir mindre effektiv desto större kodbasen är på grund av hur svårt systemet blir att beskåda som helhet.

Uppgift 7 **(2 poäng)**

Det finns flera svårigheter i samband med testdata. Nämn, och beskriv kort innebörden av, två vanliga problem/svårigheter som ofta förknippas med hantering av testdata.

- Att skapa en lagom stor uppsättning av testdata.
- Att skapa testdata som är produktionslik.
- Testdata förstörs när det används i testerna vilket gör det svårt att få tillgång till tillförlitlig testdata.

Uppgift 8 **(2 poäng)**

Nämn två vanliga kategorier av risker i samband med test och beskriv vilken karaktär/typ av risker respektive kategori handlar om.

Tekniska: Risker kopplade till teknik, som komplexitet eller tekniska fel.
Projektorganisatoriska: Risker inom projektledning, som tidsplanering eller budgetöverskridande.
Linjeorganisatoriska: Risker i linjeorganisationen, som resurskonflikter eller bristande stöd.
Marknadsrelaterade: Risker kopplade till marknaden, som förändringar i efterfrågan eller konkurrens.
Kundrelaterade: Risker kopplade till kundens krav, engagemang eller förväntningar.
Externa: Risker från omvärlden, som lagändringar eller naturkatastrofer.
Projektrisker: Risker som påverkar projektets framgång, t.ex. förseningar eller fel leveranser.

Produktrisker: Risker kopplade till produkten, som kvalitet eller funktionalitet.
Humanrisker: Risker kopplade till individer, som brist på kompetens eller sjukfrånvaro.
(välj 2 av dessa om det kmr på tentan)

Uppgift 10 (3 poäng)

Nämn tre fördelar med att använda testfall (var tydlig med respektive fördel).

Strukturerat: Testfallen är tydligt definierade och organiserade, vilket gör testprocessen mer systematisk och lätt att följa.
Bra grund för felrapporter: Eftersom testfallen är väl dokumenterade kan eventuella avvikelser enkelt identifieras och rapporteras med specifika detaljer om var och hur felet inträffade.
Lätt att byta testare: En strukturerad testspecifikation gör det enkelt att överlämna tester till olika personer, eftersom alla steg och förväntade resultat är tydligt dokumenterade.
Bra grund för automatisering: Testfall kan användas för att skapa automatiserade tester, vilket ökar testeffektiviteten och gör det lättare att återanvända tester för framtida versioner av programvaran.

Uppgift 11 (4 poäng)

Vilka rubriker ingår ofta i en testrapport?

Unik identifierare (ID)
Inledning
Sammanfattning
Avvikelser
Bedömning av testernas omfattning
Sammanfattning av resultat
Utvärdering
Erfarenheter
Sammanfattning av aktiviteter
Godkännande

Uppgift 13 (4 poäng)

När skall man använda informell respektive formell granskning?

Informell granskning: Informell granskning används vid tidsbrist eller om dokumentet som granskas är mindre viktigt. - Informell granskning kan också användas som förgranskning till en mer formell granskning i syfte att beta av många småfel på kort tid (ex. stavfel och grammatiska fel). Billigaste sätt att uppnå viss nytta och att hitta fel.
Formell granskning: Formell granskning följer en dokumenterad granskningsprocess som exempelvis beskriver att det ska finnas möte, formella roller, krav på förberedelser samt mål. Ingen tidsbrist.

Uppgift 14 (4 poäng)

”Testa tidigt” och ”Test är kontextberoende” är två grundläggande principer inom test. Förklara vad ovanstående två principer innebär. Ange också ytterligare två grundläggande principer inom test och beskriv kort vad de innebär.

Testa tidigt: Tidig testning identifierar fel snabbare och minskar kostnaderna för att åtgärda dem.
Test är kontextberoende: Testmetoder och mål beror på typen av system och användningsområde.
Fullständiga test är omöjliga: Det går inte att testa allt; prioritering krävs för att täcka de viktigaste områdena.
”Absence of errors” - fallacy: Att inga fel hittas betyder inte att systemet uppfyller alla krav eller är användbart.
(Tillhör de 7 principerna)

Uppgift 16 (10 poäng)

Det finns många fält som kan användas vid beskrivningen av testfall och felrapporter. Beskriv kortfattat för följande fält vad de innebär och ange även för respektive fält om de vanligtvis används för testfall och/eller felrapporter. De fem fält som skall behandlas är: ID, Prioritet, Allvarlighetsgrad, Förväntat resultat, och Status.

ID: Används i båda fallen och med samma syfte, att identifiera respektive testfall eller felrapport
Prioritet: används i båda fallen för att visa på hur angeläget det är att få något gjort eller åtgärdat. För testfall visar det i vilken ordning de ska prioriteras vad gäller att utföras och för felrapport visar det på hur brådskande det är att få felet åtgärdat.
Allvarlighetsgrad: finns endast i felrapport och används för att visa hur allvarligt det framfallna felet påverkar systemet
Förväntat resultat: används i både testfall och felrapporter och beskriver det resultat som testaren förväntar sig från testfall genomförandet.
Status: finns endast i felrapporter och visar vad som händer med felet just nu, exempelvis om felet är nytt eller om det är klart för omtest.
Vet ej om detta är helt rätt(dubbelkolla om ni vill)

Uppgift 17 (9 poäng)

Beskriv de grundläggande stegen för en granskningsprocess.

Planering = I detta steg definieras syftet med granskningen, vilka dokument som ska granskas, och vilka deltagare som ska involveras. En plan skapas som inkluderar tidpunkter, ansvarsfördelning och kriterier för att avgöra om materialet är redo att granskas.
Start = Här introduceras granskningen för alla inblandade. Dokumenten distribueras till granskarna, och mötets mål, omfattning och regler för processen kommuniceras tydligt. Rollerna klargörs också.

Individuella förberedelser = Varje deltagare granskar materialet på egen hand, identifierar potentiella fel, oklarheter eller avvikelser från specifikationer. Observationerna dokumenteras och förbereds för att diskuteras under granskningsmötet.
Granskningsmöte = Deltagarna diskuterar de individuella observationerna. Fokus ligger på att identifiera och dokumentera fel, inte att lösa dem direkt.
Uppföljning = Efter mötet följs eventuella åtgärder upp. Författaren av materialet reviderar det baserat på återkopplingen, och moderatorn kontrollerar att ändringar implementerats enligt rekommendationerna. Om det behövs kan en ny granskning planeras

Uppgift 18 (10 poäng)

Nämn fyra huvudsakliga framgångsfaktorer som ofta är avgörande för ett lyckat testarbete och förklara effekterna av respektive faktor.

Påbörja test tidigt: Börja testa redan i krav- och designfasen för att tidigt upptäcka och rätta till fel.
Testa under hela utvecklingsprocessen: Fortsätt testa genom hela projektet för att säkerställa kvalitet, snabbt hitta och åtgärda fel och begränsa kostnader för fel.
Struktur i testarbetet: Använd en tydlig och väldefinierad testprocess för att säkerställa systematisk och effektiv testning.
Arbeta metodiskt för att kvalitetssäkra systemet: Följ standardiserade metoder och tekniker för att konsekvent säkerställa hög kvalitet i systemet.

Tenta 3

Uppgift 1 (2 poäng)
Vem eller vilka (vilken roll) genomför vanligtvis integrationstest och vad testas vid ett sådant test i ett systemutvecklingsprojekt?

En utsedd utvecklare.
Testar integration av komponenterna inom ett system → Fel i komponenternas kommunikations-gränssnitt. Utförs inte alltid, beroende på projektets storlek.

Uppgift 2 (2 poäng)
Beskriv skillnaden mellan testning och begreppet kvalitetssäkring.

Att testa ett system i syfte att hitta fel är en del av att kvalitetssäkra arbetet. Testning syftar till att hitta fel i systemet medan kvalitetssäkring handlar om mer än bara testning och inkluderar bland annat kravhantering, projektledning och driftsättning. Med kvalitetssäkring strävar man mot att göra systemet rätt från början.

Uppgift 3 (2 poäng)
Vad är de negativa konsekvenserna förutom missnöjda kunder och användare av att inte implementera testning i ett systemutvecklingsprojekt?

Stora kostnader för utveckling: Fler resurser krävs för att rätta till problem som hade kunnat upptäckas tidigare i processen.
Högre kostnader för förvaltning: System som inte är ordentligt testade kräver mer underhåll och löpande korrigeringar, vilket ökar förvaltningskostnaderna.

Uppgift 4 (2 poäng)
Vad kännetecknar iterativ testning?

I iterativa utvecklingsmetoder bygger man en liten bit av systemet och testar denna bit. I nästa iteration utökas systemet med ytterligare funktioner och de funktioner som tagits fram i tidigare iterationer förfinas. Sedan testas nästa del osv.

Uppgift 5 (2 poäng)
Nämn två regler för tidsuppskattning.

Estimera nödvändig tid och definiera start och slutkriterier ELLER

Små aktiviteter är lättare att estimera än stora aktiviteter.

Enkla aktiviteter är lättare att estimera än komplexa aktiviteter.. (((()))) (oklart om detta är rätt eller vilket som är rätt!)

Uppgift 6 (2 poäng)
Varför finns det inga felfria system?

Inga system är felfria eftersom de ofta är komplexa och det är omöjligt att testa alla möjliga scenarier. Mänskliga fel, ofullständig testning och förändringar under utvecklingen kan introducera nya problem. Dessutom kan användare interagera med systemet på oväntade sätt, vilket kan leda till fel som inte förutsetts. Detta gör att det alltid finns en risk för fel, även i noggrant utvecklade och testade system. Fel hopar sig också ofta, hittar man fel och löser det kan det bli fel annanstans.

Uppgift 8 (2 poäng)
Vad innebär avbrytandekriterier respektive återupptagandekriterier i en testplan?

Avbrytandekriterier: Kriterier för att avsluta tester på ett onormalt sätt. Kan handla om att det inte finns tillräckligt med resurser eller att en stor mängd fel har framkommit av systemet.

Återupptagandekriterier: Kriterier som ska vara uppfyllda för att återuppta testerna. Regressionstester har genomförts och resultatet visar att problemen är åtgärdade.

Uppgift 9 (2 poäng)
Redogör fördelarna och nackdelarna med ad hoc testning.

Fördelar:

- Är gynnsamt vid nyutvecklade program, då dessa oftast innehåller mer fel (såsom felstavning etc).
- Ett bra sätt att introducera nya testare dels för test men också för att bekanta sig med ett system.

Nackdelar:

- Inga testfall
- Man vet inte hur mycket testande som är kvar, då det inte finns ett visst antal testfall.
- Det är svårt att veta om alla funktionaliteter är testade.
- Det är svårt att minnas vad som orsakade ett fel.

Uppgift 10 (3 poäng)
Ange tre stoppkriterier för när det är färdigtestat.

1. *Alla testfall är körda.*
2. *Inga kvarvarande öppna fel med högsta allvarlighetsgrad får förekomma*

3. *90% kodsatstäckning är uppnådd.*

Uppgift 11 (3 poäng)
Förklara konceptet statisk analys.

Statisk analys = Används i syfte att underlätta granskning av programkod. Hittar fel utan att exekvera programmet. Innebär att identifiera fel utan att köra programmet. Exempel på analys för viss typ av brister;

- Avvikelser från utvecklingsstandard - Exempelvis namngivning av variabler.
- Död kod - Kod som inte kan köras någonsin, då koppling för start saknas. Svårt att hitta manuellt
- Oändliga slingor - Felaktiga villkor som aldrig kan inträffa.
- Beräkning av komplexitet - Kan hitta kod som är mer komplex och därmed testa denna mer

Uppgift 12 (3 poäng)
Förklara innebörden och skillnaden mellan White-box och Black-box testning samt vilken kategori av testning dessa två tekniker tillhör.

White-box testning White-box innebär att testaren känner till den inre konstruktionen av koden och testar exempelvis kodvillkor, kodsatser och loopar i syfte att identifiera fel i koden. Man strävar efter att testa med så hög kodtäckning som möjligt för att täcka testandet av hela programmet. White-box tester används främst under komponent, integration och systemtester och uteslutas helt under acceptanstester. Utöver detta använder man oftast diverse verktyg för att utföra denna sortens tester.

Black-box testning Vid black-box testning har testaren inte någon kännedom om systemets inre konstruktion, koden exempelvis är för testaren okänd. Som underlag för testerna kan man använda sig av krav- och designspecifikationer samt själva systemet. Black-box används främst vid acceptanstest men förekommer på alla testnivåer.

Båda testdesignsteknikerna är dynamiska och testar programmet medan det exekverar.

Uppgift 13 (5 poäng)
Vad är målet/målen med granskning?

Målen med granskning:

Verifiering
Validering
Konsensus (Nå samsyn)
Förbättringar (Hitta förbättringsförslag)
Felsökning (Hitta fel i dokumentation och kod innan felen byggs in i programmet).

Uppgift 14 (4 poäng)
Redo gör hur kurvan för detaljeringsgraden i V modellen förhåller sig ifrån start till slut i ett utvecklingsprojekt där testning genomförs.

I V modellens övre del (där kraven tas fram och acceptanstester genomförs) är det få detaljer. Ju längre ner i modellen, desto mer detaljerade blir dokumenten och aktiviteterna. I komponenttester genomförs detaljerade tester på varje enskild komponent medan man i acceptanstester genomför tester på en övergripande nivå. ((((()))))

Uppgift 15 (6 poäng)
Riskhantering är en proaktiv arbetsmetod som används i samband med testplanering. Syftet med riskhantering är att identifiera inre och yttre hot mot projektets framgång. Ange aktiviteterna i riskhanteringanalysen och beskriv dem kort.

1. **Identifiera risker** (med hjälp av workshop eller brainstorming)
2. **värdera risker**
   - ↣ Sannolikhet att risk inträffar
   - ↣ Konsekvens om risk inträffar

     Man sätter ett värde mellan 1-5 på **sannolikheten** att risken inträffar.
     Man sätter ett värde mellan 1-5 på **konsekvensen** av att risken inträffar.
     Riskvärdet är multipeln mellan sannolikhet och konsekvens (1-25, där 25 är värst).
3. **Dokumentera resultatet**
   - ↣ Risklista
4. **Planera för åtgärder**

Uppgift 16 (10 poäng)
Vad är de sju principerna om testning och utveckla dess innebörd?

1. **Test påvisar att fel finns**: Testning visar att det finns fel, men kan aldrig bevisa att systemet är felfritt.
2. **Fullständiga test är omöjliga:** Det går inte att testa allt; prioritering krävs för att täcka de viktigaste områdena.
3. **Testa tidigt**: Tidig testning identifierar fel snabbare och minskar kostnaderna för att åtgärda dem.
4. **Fel hopar sig ofta**: Fel tenderar att samlas i vissa funktioner eller delar av systemet.
5. **Anpassa/variera test**: Teststrategier bör anpassas för att täcka olika scenarier och användningsfall.
6. **Test är kontextberoende**: Testmetoder och mål beror på typen av system och användningsområde.
7. **”Absence of errors” - fallacy**: Att inga fel hittas betyder inte att systemet uppfyller alla krav eller är användbart.

Tenta 4

Uppgift Redogör vad positiva och negativa tester innebär och vad dess skillnad i syfte utgör.

Vid görandet av **positiva tester** testar man systemet under normala omständigheter. Testdatan utgörs av korrekt data. Ex. om vi ska testa att mata in en månad i ett fält är det lämpligt att skriva ex. 5 som är ett giltigt värde.
Vid **negativa tester** vill man se till systemets felhantering och hur det reagerar under "onormala" omständigheter. Som fortsättning på det tidigare exemplet kan ex. på testdata vara 15 som inte är ett giltigt värde

Uppgift På vilket sätt är statisk analys och granskning relaterat?

De båda utgör huvudområden i statisk testning. Statisk analys är ett verktyg för att analysera kod. Granskning handlar om att manuellt leta efter fel i kod och dokumentation.

Uppgift Vad är gränsvärdesanalys och vad är resonemanget för gränsvärdesanalys?

Gränsvärdesanalys: Tester som fokuserar på gränsvärden för indata, eftersom problem ofta uppstår vid dessa punkter (t.ex. minimum och maximum värden som systemet ska hantera).
Det finns två typer av gränsvärden:

Giltigt gränsvärde = ett värde på den tillåtna sidan av gränsen. (över- eller underskrider ej)
Ogiltigt gränsvärde = är ett värde på den otillåtna sidan. (över- eller underskrider)

Har man exempelvis att man ska mata in datum i ett system med en gräns på 1-12 blir därför exempel på giltigt gränsvärde → 1, 2 och 11 och ogiltigt blir —> 0 och 13 då vi bara vill ha värden mellan 1-12. Samtliga nämnda gränsvärden ska testas. Vid gränsvärdesanalys utgår man från att de ogiltiga gränsvärdena har större sannolikhet att vara fel.

Uppgift Förklara begreppet validering.

Att validera kallas det när man säkerställer att något är giltigt eller korrekt, något som ligger på kundens ansvar att göra. Inom testning innebär detta mer specifikt att jämföra en funktion med det syfte som funktionen ska uppfylla. Man kollar därför om systemet man byggt är rätt system som kunden eftersträvar.

Uppgift Förklara begreppet verifiering

Att verifiera innebär att man kollar om man har byggt systemet rätt, dvs relaterar systemets funktioner till de krav som kunden ville ha och kollar om de har uppfyllts. Detta gör oftast utvecklarna.

Uppgift När skrivs testrapporten, vad är syftet med rapporten och ge två exempel på innehåll i rapporten?

Testrapporten skrivs i slutet av testarbetet vid tillfället då alla testfall är genomförda och resultatet dokumenterat. Testrapporten är en sammanställning av testarbetet i sin helhet och innehåller bland annat erfarenheter som vill skickas med till fortsatt testarbete och en summering av antal fel som hittats och är kvarstående.

Uppgift Vad är syftet med såpoperatester?

Att försöka hitta så udda kombinationer som möjligt för att genom det identifiera fel. Innebär att man testar systemet utanför den normala användningen för att hitta varianter och kombinationer som inte tidigare är testade

Exempel: Ett exempel för SJ

1. Boka en resa tur och retur mellan Stockholm och Örebro.
2. Avboka resan
3. Boka resan igen
4. Ändra returresan till Gävle istället för Stockholm
5. Tåget blir inställt
6. Boka om till ett senare tåg

Slutar med att han får tillbaka 15 kr trots att resan till Gävle innebar att han fick åka till Stockholm och byta till ett tåg till Gävle. Ska antagligen inte vara billigare att få åka en längre sträcka.

Uppgift Vad kännetecknar test i agila utvecklingsmetoder?

Testa tidigt, korta iterationer (ca 4 veckor) av hög kvalitet och sparsamt med dokumentation. Mycket kundmedverkan.

[Testdisciplinen inom XP]

Testdriven utveckling (TDD): Först skriver man test för att beskriva hur koden ska fungera, och sedan skriver man koden för att få testet att gå igenom.
Enhetstester: Man testar små delar av koden för att se till att allt fungerar som det ska.
Kontinuerlig integration: Man slår ihop och testar koden ofta för att säkerställa att systemet alltid är uppdaterat och fungerar.
Parprogrammering: Två utvecklare jobbar tillsammans och skriver både kod och tester, vilket gör att de kan hitta problem snabbt.
Regressionstester: Man testar att nya ändringar inte orsakar problem med funktioner som redan fungerar.

Uppgift Beskriv arbetsflödet för scrum metoden.

Scrum-processen baseras sig på korta iterationer (sprints) där man utgår från en backlog som innehåller de krav som ska uppfyllas. Man har dagliga scrummöten där man fördelar arbetet samt diskuterar hur det går. Scrum är därför en agil metod.

Uppgift Vad är de negativa konsekvenserna av misslyckat testarbete?

- Driftsättning försenas, ju närmare den felen hittas desto mindre tid att åtgärda.
- Användarna hittar felen istället för testarna.
- Följdfel i andra delar av systemet när ett fel rättas
- Utvecklare får arbeta övertid och belastningen på supportavdelningen ökar.
- Bristande kvalité= bristande förtroende för systemet eller företaget.

Uppgift Beskriv två stycken exempel på test inom testnivån komponenttest.

Test av en enskild funktion exempelvis att; Spara en kund, Validera input

1. **Funktionstest av metod** – Testar att en metod returnerar korrekt resultat, t.ex. att summan av 3 och 5 blir 8.
2. **Databasinteraktionstest** – Verifierar att en komponent hämtar rätt data från databasen och hanterar fel, t.ex. tomma svar.

Uppgift Testdesigntekniker kan delas in i två huvudsakliga kategorier beroende på om de avser att användas på ett körande program eller inte. Nämn dessa.

Statisk testning: Innebär att identifiera fel utan att köra programmet. Det handlar om att granska kod, design, och dokumentation för att hitta misstag och brister innan något körs. Detta kan göras genom kodgranskningar eller inspektioner där utvecklare eller testare går igenom materialet för att upptäcka problem tidigt.

Dynamisk testning: Exekvera programmet för att hitta fel. Innebär att faktiskt köra programmet för att se hur det fungerar i praktiken och hitta buggar genom att testa funktionaliteten (whitebox, blackbox)

Uppgift Vad kännetecknar ett bra testfall?

- En tydlig rubrik som berättar vad testfallet syftar till att testa.
- Tydliga teststeg som en annan testare kan följa utan problem.
- Förväntat resultat som visar vad som förväntas när teststegen utförs.
- Förberedelser som visar vad som behöver utföras innan testfallet kan utföras.

Uppgift Beskriv de huvudsakliga fyra dokument delarna som ska ingå i en testspecifikation.

Testdata: Specificerar de uppgifter eller indata som används under testerna för att simulera verkliga användarscenarier och kontrollera systemets funktionalitet.
Förberedelser: Innebär de aktiviteter som måste genomföras innan testningen, som att konfigurera testmiljöer och förbereda testdata.
Testfall: Beskriver detaljerade steg för att testa en viss funktion, inklusive förväntade resultat och vilken del av systemet som testas.
Återställning: Efter testning ska systemet återställas till sitt ursprungliga tillstånd genom att rensa testdata och återskapa tidigare konfigurationer.

Uppgift Testverktyg kan delas in i ett antal huvudsakliga kategorier beroende på syftet. Ange fyra av dessa.

Kravhanteringsverktyg: Verktyg för att dokumentera och spåra krav under utvecklingsprocessen (t.ex. Jira, IBM Engineering Requirements).
Felrapporteringsverktyg: Verktyg för att dokumentera och följa upp buggar och fel (t.ex. Bugzilla, Jira).
Testhanteringsverktyg: Verktyg för att planera, organisera och spåra tester (t.ex. TestRail, HP ALM).
Versionshanteringsverktyg: Verktyg för att hantera och spåra kodändringar (t.ex. Git, SVN).
Testautomatiseringsverktyg: Verktyg för att automatisera tester (t.ex. Selenium, QTP).
(Finns många fler)

Uppgift Till beskrivningen av en bristande funktion i en felrapport kan både prioritet och allvarlighetsgrad anges. Beskriv innebörden av dessa begrepp.

Prioritet avgör när i tiden ett fel ska åtgärdas och hur brådskande det är.
Allvarlighetsgrad är till för att visa hur allvarligt felet påverkar utvecklingen av systemet eller drift och utveckling av komponent.

Uppgift Sättet att bedriva testning av IT-system kan variera beroende på vad för slags systemutvecklingsmetod som används. Beskriv testinsatsen för sekventiella utvecklingsmetoder och ange två vanliga problem med denna ansats.

Den sekventiella utvecklingsmetoden utgår från en sekvens av aktiviteter, man påbörjar arbetet med en analys fas, sedan en designfas, implementationsfas och slutligen en testfas. Problematiken med denna utveckling metod är att man inte arbetar i iterationer och därför skapas hela programmet innan den testas. Testarna får då en massa fel som måste åtgärdas samtidigt som felen upptäcks väldigt sent och kostar mer att åtgärda.

Uppgift Det finns många testdesigntekniker som kan användas för att utforma testfall som kan genomföras av en testare utan programmeringskunskaper. Ange fem stycken av dessa och beskriv dem.

Utan programmeringskunskap ligger lämpliga testdesigntekniker under Black box testing som fokuserar på att testa systemets funktionalitet utan att ta hänsyn till kod.

- **Positiva tester**: testa programmet i normalanvändning, exempelvis lägga till en kund, ta bort en kund etc.
- **Negativa tester**: testa programmets felhantering, exempelvis skriva in siffror i fält där det ska vara bokstäver.
- **Extremtester**: syftet är att förstöra systemet exempelvis genom att dra ut nätverkskabeln eller slänga datorn i väggen.
- **Gränsvärdesanalys**: testa de värden som är både innan och efter respektive gränsvärde.
- **Såpoperatester**: man testar systemet utanför den normala användningen med olika kombinationer av varianter för att hitta oförväntade resultat.

Uppgift Testprocessen delas vanligtvis in i de tre faserna planering, genomförande och uppföljning. Nämn minst tre huvudsakliga aktiviteter i respektive fas samt beskriv kortfattat vad respektive aktivitet innebär.

Planering

1. Identifiera testområden: man läser rubriker i kravdokumenten för att identifiera områden som ska testas.
2. Granska kravdokument: man granskar kravdokumenten i syfte att se om alla krav är testbara.
3. Bedöm riskerna: man identifierar risker som kan uppstå under arbetet och som kan påverka testande negativt.

Genomförande

1. Följ testplanen: man utför de aktiviteter som specificerats i testplanen.
2. Rapportera löpande: man skriver felrapporter när ett fel upptäcks.
3. Dokumentera i en testlogg: man dokumenterar det dagliga testarbeteter och alla aktiviteter i en testlogg.

Uppföljning

1. Sammanfatta: man skriver en testrapport som sammanfattar resultat av arbetet.
2. Utvärdering: Analyserar testprocessen för att bedöma effektiviteten och om målen uppnåddes.
3. Samla erfarenheter: man lyfter de positiva och negativa lärdomar man har tagit från testarbetet för att använda dessa till nästa gång.

Uppgift Vad är skillnaden mellan testfall och testlogg?

Testfall är ett dokument som beskriver vad som ska testas med "test steg", förväntad resultat, etc.
Testlogg är att man dokumentera test arbetets aktiviteter kontinuerligt genom dagen. Underlättar om man byter användare eller om man vill se tillbaka på det som gjorts.

Uppgift En överlämning dokumenteras ofta i ett överlämnande dokument. Beskriv syften med ett överlämnande dokument.

Dokumentet beskriver vad som lämnas från en testomgång till en annan. Kan också vara från utvecklare till testare eller från leverantör till kund.

Uppgift Identifiera och beskriv de tre vanligaste stegen i ett typisk flöde för testgenomförande.

1. **Test** – första test.
2. **Omtest** – Beroende på hur snabbt eller inte ett testfall lyckats eller kvarstår. Om fel kvarstår görs fler iterationer.
3. **Regressionstest** – Försöker köra funktionella och icke funktionella tester. För att se att det som utvecklats fortfarande funkar. Man måste se att det klaras av fortfarande efter ändringar. Testar det som ändrats.

Uppgift I arbetet med testplanering finns det flera vanliga planerings misstag. Nämn fyra vanliga misstag vid testplanering och beskriv respektive misstag handlar om.

1. **Ofullständig eller bristfällig testplan** - otillräcklig dokumentation. brister i förberedelser,, riskbedömning, kriterier, tillvägagångssätt etc.
2. **Brister i kravhantering** - otydliga testmål och krav. Otydlig eller otillräcklig definition av vad som ska testas, vilket resulterar i att testningen blir ospecifik eller missar viktiga funktioner.
3. **Brist i tid- och resursplanering** - Underskattade tidsramar och resurser, vilket kan göra att tester inte genomförs ordentligt eller försenar projektet.
4. **Bristande kommunikation** - innebär att viktig information inte överförs effektivt mellan team, intressenter eller system. Detta kan leda till missförstånd, förseningar och kvalitetsbrister
5. **Brist på riskanalys** - risker identifieras inte, vilket kan reslutera i kritiska problem sent in i utvecklingen.

Uppgift Fördelar och nackdelar med testerna a) Adhoc testning, b) Checklistor, c) utforskade testning?

Adhoc = testa utan stöd av testfall eller andra testunderlag.
- svårt att upprepa ett test exakt
+ bra sätt att introducera nya testare.
Checklistor = mellanting mellan strukturerade och adhoc tester.
- svårt att skriva felrapport.
+ bra vid tidsbrist

Utforskande testning = erfarenhetsbaserad testdesignteknik utan skriftliga testfall. Testerna skapas samtidigt som testarna lär sig systemet och utför testerna.
- Testtiden blir inte nödvändigtvis kortare och kräver mycket testledning.
+ Testarna lär sig systemet snabbt vilket kan leda till bättre resultat i nästa testomgång.

Uppgift Ange tre anledningar till vad det är som gör granskning effektiv för kvalitetssäkring.

Hitta fel tidigt

1. Hittar fel i dokumentation och kod innan felen byggs in i programmet (gör granskning kostnadseffektivt)

Förbättringsförslag

2. Dessutom hjälper granskning med att hitta förbättringsförslag innan utveckling.

Samsyn

3. Granskningar samlar teammedlemmar för att gemensamt granska krav, design och tester. Detta skapar samsyn och minskar risken för missförstånd och säkerställer att alla har förståelse för systemet, på dess mål och på vad som ska levereras.

Uppgift 2 **(2 poäng)**

Förklara vad funktionen är med en testdagbok.

Uppgift 3 **(2 poäng)**

Uppgift 2:

Testdagbok även kallad testlogg är en sammanfattning av viktiga händelser under dagen. Funktionen är då att testare kan gå tillbaka och se vad som gjorts tidigare vilket också är bra om man är flera testare. Samt se över viktiga händelser som skett.

2

Uppgift 3:

Dokumentgranskning har fördelar som att hitta fel, också tidiga fel som blir enklare och billigare att korrigera innan systemutvecklingen har börjat. Dessutom finns fördelar såsom att hitta förbättringsförslag och nå konsensus.

2

Uppgift 4 **(2 poäng)**

Enligt kurslitteraturen är kraven den enskilt största felkällan vid utveckling av IT-system. Ange två tänkbara felkällor vid framtagning av krav.

Uppgift 4:

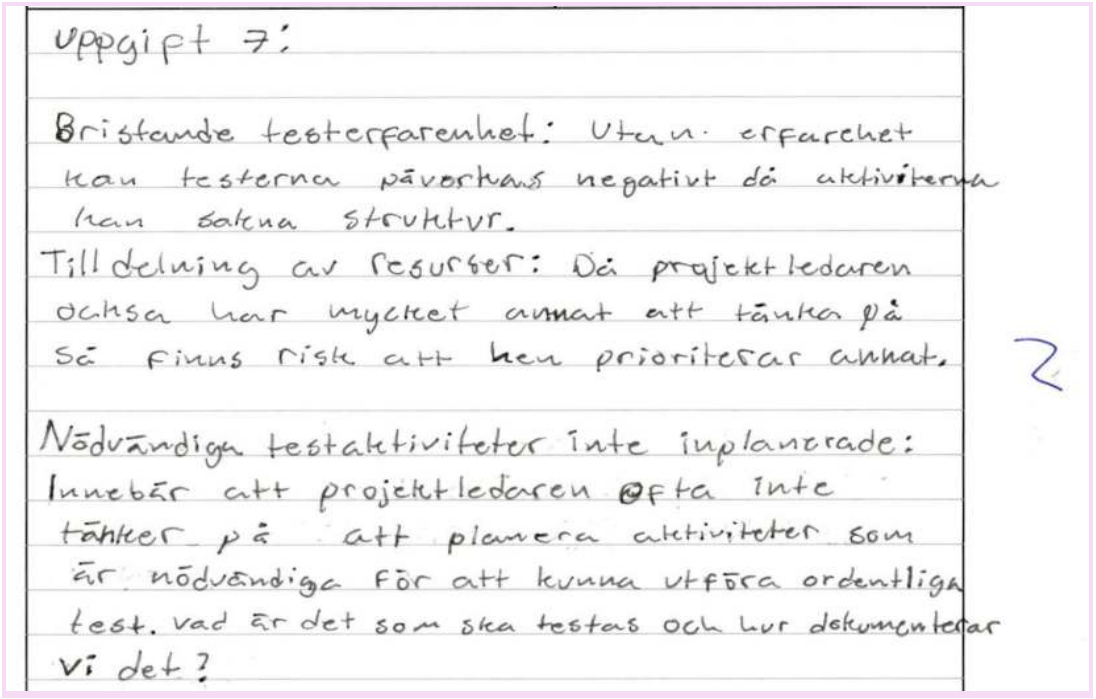
Fel personer deltar vid kravinsamling: innebär att kraven som sätts kanske inte anpassas till de som ska använda systemet.

Ingen prioritering av kraven: kraven prioriteras inte vilket leder till att krav som är väldigt nödvändiga kanske inte sätts i bruk.

2

Uppgift 7 **(3 poäng)**

Vilka risker kan finnas om projektledaren ansvarar för testaktiviteterna?



Uppgift 7:

Bristande testerfarenhet: Utan erfarenhet kan testerna påverkas negativt då aktiviterna kan saknas struktur.

Tilldelning av resurser: Då projektledaren också har mycket annat att tänka på så finns risk att hen prioriterar annat.

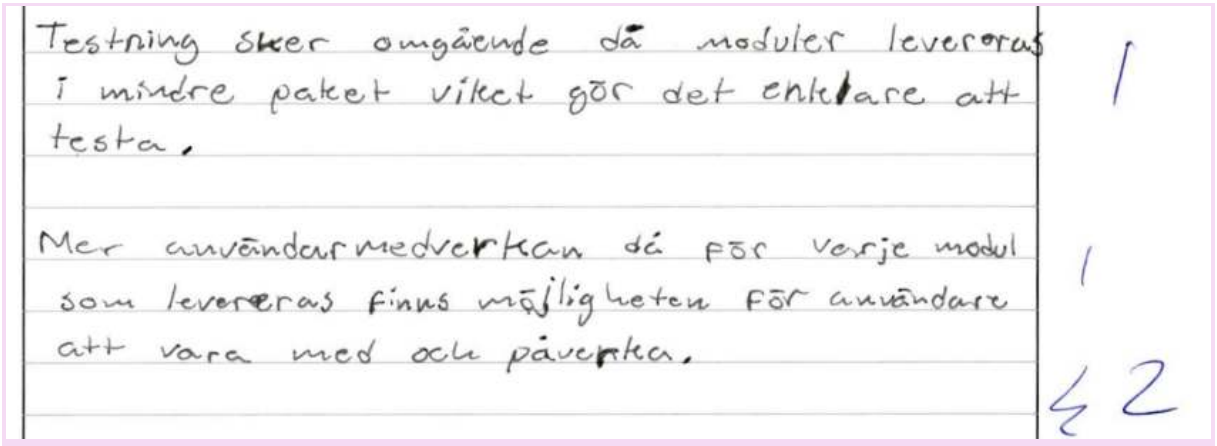
Nödvändiga testaktiviteter inte inplanerade: Innebär att projektledaren ofta inte tänker på att planera aktiviteter som är nödvändiga för att kunna utföra ordentliga test. Vad är det som ska testas och hur dokumenterar vi det?

2

(Använd denna som komp till samma fråga längre ner^)

Uppgift 8 **(3 poäng)**

Ange tre fördelar med iterativ testning jämfört med testning enligt vattenfallsmodellen.



Testning sker omgående då moduler levereras i mindre paket vilket gör det enklare att testa.

1

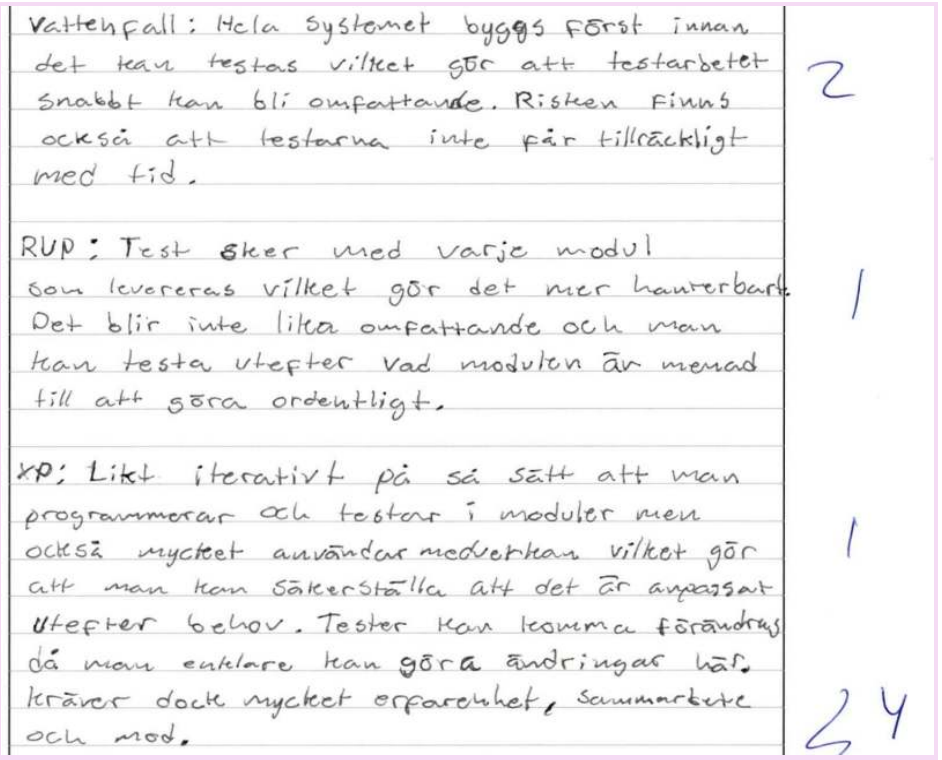
Mer användarmedverkan då för varje modul som levereras finns möjligheten för användare att vara med och påverka.

1

½ 2

Uppgift 13 **(6 poäng)**

Vi har i kursen gått igenom hur test utförs i tre olika systemutvecklingsmetoder (Vattenfall - sekventiellt, RUP - iterativt och inkrementellt och XP - agilt). Beskriv hur respektive metod påverkar hur testarbetet genomförs.



Kort förtydligande:

Metod	Beskrivning och Testning
Vattenfall	Sekventiell metod där varje fas görs steg för steg. Testning sker i slutet, vilket kan leda till sena felupptäckter.
RUP (Rational Unified Process)	Iterativ metod där systemet förbättras i faser. Testning sker iterativt för att säkerställa funktion i varje del.
XP (Extreme Programming)	Agil metod som fokuserar på kontinuerlig utveckling och snabb feedback. Testning sker löpande med testdriven utveckling (TDD) och automatisering.

Uppgift 16 **(8 poäng)**

Placera ut nedanstående typer av tester under black box-testning respektive white box-testning samt förklara skillnaden mellan black box-testning och white box-testning.

- positiva & negativa tester,
- kodsatstestning,
- looptestning,
- säkerhetstester,
- flödestester,
- extremtester,
- kodtäckningsanalys,
- villkorstestning
- såpoperatester
- kodgrenstestning

Blackbox: Innebär att man inte har någon insyn i systemets komponenter och kod, utan syftet är att testa så att systemet klarar av vardaglig hantering i form av positiva & negativa tester, stresstester mm.
Olika typer av tester i blackbox:
Positiva & negativa tester
Säkerhetstester
flödestester
Extremtester
Säpoperatester

Whitebox: Innebär att man granskar och testar kod och systemets insida. Mer back-end orienterad form av testning. Syftet är att se så kod fungerar som det ska och ingen dö kod finns.
Olika typer av tester i Whitebox:
Kodsatstestning
looptestning
kodtäckningsanalys
villkorstestning
kodgränstestning

8

Uppgift 17 **(9 poäng)**

Testdriven utveckling och automatisering av tester har blivit alltmer vanligt, bl a för effektivisera testarbetet och att både utvecklare och testare ska fokusera test och kvalitet.

a) Förklara kortfattat vad testdriven utveckling innebär (2p).

b) Beskriv kortfattat två huvudsakliga begränsningar med manuell testning. (2p).

c) Det finns flera fördelar med automatiserad testning; förklara innebörden av tre fördelar med automatiserad testning (3p).

d) Det finns också flera utmaningar med automatiserad testning; förklara innebörden av två huvudsakliga utmaningar med automatiserad testning (2p).

A) Testdriven utveckling (TDD) innebär att tester skrivs innan kod implementeras. Kod skrivs därefter för att klara testerna, vilket säkerställer kvalitet och funktionalitet från början.

B)

1. **Tidskrävande** – Manuell testning tar lång tid, särskilt vid repetitiva tester.
2. **Risk för mänskliga fel** – Misstag kan uppstå på grund av trötthet eller ouppmärksamhet.

C)

1. **Snabbhet** – Testerna kan köras snabbt och upprepas automatiskt.
2. **Tillförlitlighet** – Minskar risken för mänskliga fel.
3. **Skalbarhet** – Lätt att testa stora kodbaser och komplexa system.

D)

1. **Initial kostnad** – Kräver investering i verktyg och tid för att skapa testskript.
2. **Underhåll** – Testskript måste uppdateras vid förändringar i koden.

Uppgift 6: beskriv i vilket avsnitt i en testplan det går att hitta information om hur fel ska rapporteras i testarbetet, för de testare som ska genomföra testerna:

- I avsnittet ”Tillvägagångsätt” beskrivs hur felrapportering ska gå till. Hänvisa till ett separat dokument om felrapporteringsprocessen är invecklad.

Uppgift 7: Vilka risker kan finnas om projektledaren ansvarar för testaktiviteterna?

- Intressekonflikt: Projektledaren har oftast ett starkt intresse av att projektet ska slutföras i tid och inom budget.
- Brist på specialkunskap: Testning är en specialiserad uppgift som kräver specifika kunskaper och erfarenheter.
- Resurstilldelning: Projektledaren måste balansera olika resurser och prioriteringar. Om de också ansvarar för testaktiviteter kan testning få lägre prioritet jämfört med andra projektuppgifter. Särskilt när projektet är under tidspress.

Uppgift 11: Ett viktigt fält i ett testfall är ”Förväntat resultat”. Hur bär beskrivningen formuleras för detta fält? Ge exempel på en bra beskrivning och en mindre bra beskrivning för ”Förväntat resultat”

- Dåligt formulerat: ”Resultatet ska bli som förväntat”
- Bra formulerat: ”Kunden sparas. Kontrollera att det går att söka fram kunden med sökfunktionen”

Uppgift 18: Testdriven utveckling har blivit alltmer vanligt, bl.a. för att förbättra komponenttesterna.

A) Förklara kortfattat vad testdriven utveckling innebär

- Testdriven utveckling innebär att man skriver tester först innan man skriver koden.

B) Testdriven utveckling beskrivs vanligtvis med tre huvudsakliga aktiviteter, förklara de tre huvudsakliga aktiviteterna

- Skriva tester som initialt misslyckas.
- Skriva kod tills testerna funkar.
- Förfina koden men behålla funktionaliteten (refaktorisera)

C) Det finns flera fördelar med testdriven utveckling; Förklara innebörden av fyra fördelar med testdriven utveckling.

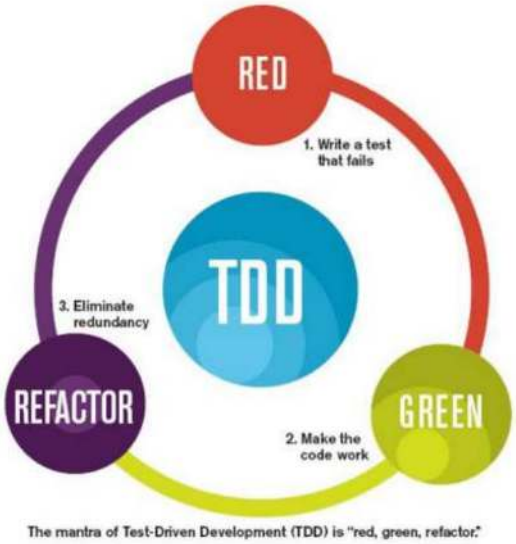
- Förbättrad kodkvalitet
    - Eftersom utvecklare skriver tester innan de implementerar funktioner, tvingas man tänka på krav och design från början.

- Tidigare upptäckt av buggar
    - Genom att skriva test först kan många buggar upptäckas och åtgärdas tidigt i utvecklingsprocessen.

- Underlättar refaktorisering
    - Eftersom det finns testpaket som täcker de befintliga funktionaliteten kan utvecklare refaktorera och förbättra koden med större säkerhet

- Ökad förtroende för koden
    - Med en stor uppsättning automatiserade tester kan utvecklare ha större förtroende för att ändringar och nya funktioner inte introducerar regressionsfel.

Red – Green - Refactor



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

1. Vi börjar med att skapa en uppsättning tester som initialt misslyckas
2. Vi skriver kod och testar densamma tills alla tester lyckas
3. Vi omstrukturerar och förfinar koden med bibehållen funktionalitet (denna process kallas refactoring)

Uppgift 5: På vilket sätt kan standarder påverka hur testerna genomförs?

- Standarder är till för att undvika missförstånd, skapa enhetliga och transparenta rutiner. Detta påverkar testerna positivt då det finns ett tydligt och strukturerat sätt att göra testerna och dokumentationen på.

Uppgift 11: Vad innebär den så kallade S-kurvan och vad används den till i samband med test

- S-kurvan är ett mätetal som visar antalet felrapporter. Innebär att i början av testarbetet är det sannolikt att ett stort antal fel hittas, ju längre tid som går desto färre fel för varje dag hittas.

Uppgift 3: Beskriv vad man kan utgå ifrån om det inte finns några formulerade krav

- Finns det inte några formulerade krav är det viktigt att man är erfaren och kan utgå ifrån det, men också ha mycket användarmedverkan i utvecklingen av systemet så man hela tiden kan få feedback på vad som önskas.

Uppgift 4: Ange två olika typer av mätetal som kan användas för att övervaka testprocessen

- Statistik om felrapporter: Med hjälp av S-kurvan.
- Kravtäckning: Mäta testfall till varje krav. Exempelvis spårbarhetsmatris.

Uppgift 8: Förklara vad det är som avgör när testerna ska avslutas i ett testarbete.

- Slutkriterierna i testplanen

1. *Alla testfall är körda.*
2. *Inga kvarvarande öppna fel med högsta allvarlighetsgrad får förekomma*
3. *90% kodsatstäckning är uppnådd.*

Uppgift 17: Kravhanteringsprocessen Stjärnan är ett praktiskt exempel på en tillämpbar kravhanteringsprocess. Stjärnan består av sex steg: samla in, strukturera, prioritera, dokumentera, kvalitetssäkra och förvalta. Din uppgift är att beskriva vad som definierar respektive steg.

Samla krav - samlar in krav som kommer att göra underlag för arbete. Sätt och tekniker för att samla in krav.

SRS för gammalt system: *Analys av kravspecifikationen (Software Requirements Specification) för det befintliga systemet för att identifiera vad som ska förbättras eller behållas.*

Verksamhetsmodellering: *Kartlägga verksamhetsprocesser och flöden för att identifiera krav som speglar affärsbehov och mål.*

Intervjuer: *Samtal med intressenter och användare för att få insikt om deras behov, förväntningar och problem med det nuvarande systemet.*

Krav-workshop: *Gruppmöten med intressenter för att diskutera och dokumentera krav gemensamt i en strukturerad process.*

Prototyp-byggande: *Skapa enkla versioner av systemet eller funktionerna för att få feedback och bättre förstå användarnas krav.*

Scenarier/Storyboards: Visualisering av användarscenarier eller arbetsflöden för att tydliggöra hur systemet ska användas.

Brainstorming: Gruppdiskussioner för att generera idéer och krav på ett kreativt sätt utan initiala begränsningar.

Användningsfallsmodellering: Dokumentera användarnas interaktion med systemet i olika scenarier för att definiera funktionella krav.

Strukturera krav - Strukturering av krav är en aktivitet som pågår kontinuerligt. Aktiviteten går ut på att skapa struktur som är lätt att överblicka och förvalta. *Metoder*: Använd hierarkiska kategorier, grupperingar eller verktyg som kravhanteringssystem för att hålla krav lättillgängliga och spårbara.

Prioritera krav - Fokusera på det viktigaste, Hitta hög- och låg-prioriterade krav, Implementation av rätt krav i rätt ordning, Spara tid och pengar.

Dokumentera krav - Dokumentation säkerställer att alla krav är tydligt definierade, spårbara och begripliga för utveckling och test.

Kvalitetssäkra krav - En ständigt fortlöpande aktivitet som syftar till att säkerställa att de dokumenterade kraven dels beskriver rätt egenskaper och dels beskriver dem på rätt sätt. *Validering* – bygger vi rätt system? *Verifiering* – bygger vi systemet rätt?

Förvalta krav - Förvaltning av krav handlar om att säkerställa att förändringar hanteras strukturerat och effektivt under hela projektet. spårbarhetsmatris

Uppgift 18 Beskriv IEEE 829-2008

IEEE 829-2008 är en standard för att strukturera och dokumentera testprocesser, inklusive testplaner, testfall och testrapporter, för att säkerställa tydlighet och spårbarhet.

Struktur för hela testprocessen:

- Bryter ner den övergripande testplanen i mindre delar för att underlätta genomförandet.
- Testarbetet delas upp i olika nivåer, såsom enhetstestning och systemtestning.
- Separata dokument skapas för planering och testfall, vilket ger tydlig struktur.

- **Kritik =** Kan **inte vara flexibel**, leder till **slöseri med tid** och kan ha **fel fokus** om det inte används korrekt.
- **Fördelar = Bra checklista** för att säkerställa att alla viktiga delar täcks - om den används **flexibelt**.