Hemuppgift 1DV404

Deluppgift 1 – Processmodeller (250 ord)

a)

Syftet hos organisationer är att skapa upprepbara, förutsägbara och överförbara mjukvaruutvecklingsprocesser där alla arbetar mot samma mål. Med en utvecklingsprocess är risken mycket mindre att processen kostar mer än beräknat, chansen är större att processen håller tidsramarna, att kvalitén blir bättre och de blir enklare att hantera. Chansen att lyckas ökar.

b)

Roller; d.v.s **vem** som gör vad, exempelvis en programmerare. Aktiviteter; **vad** någon/något gör, exempelvis när en programmerare skapar en klass. Artefakter; **det** som det görs något med, exempelvis en kravspecifikation.

c)

Det är skillnad på hur normativa processmodellerna är, d.v.s hur styrd arbetet är.
UP är en stor processmodell med fastare föreskrivna roller och artefakter som ska följas och hållas, som exempelvis ett recept man kan följa i kokbok som beskriver hur man gör. SCRUM å andra sidan används av agila utvecklare och är inte lika strikt normativ, har få bestämda roller, artefakter och aktiviteter jämfört med Unified Process. SCRUM processen ger mer frihet då de endast ställer krav på hur arbetet ska organiseras medan UP ställer mera krav på individerna när det gäller mjukvaruutvecklingen. De mindre normativa utvecklingsprocesserna som SCRUM kräver mer erfarenhet och kompetens av personal där de kan komma att göra varandras arbetsuppgifter, medan i UP är personal i sin specifika yrkesroll och har tydligare beskrivningar på vilka arbetsuppgifter ska göras och hur. SCRUM har också exempelvis bara tre typer av roller samt tre typer av artefakter vilka är förutbestämda i jämförelse med UP där de är många fler.

Deluppgift 2 – Kravhantering (250 ord)

a)

Funktionella och icke funktionella krav. Funktionella krav beskriver just krav på funktionalitet som ska finnas i systemet, exempelvis att ett hemlarmsystem ska skicka sms till ägaren vid inbrott. Icke funktionella krav betonar på kvalitet snarare än funktionalitet, exempelvis systemprestanda; hur lång tid det maximalt får ta för systemet att utföra en funktion. Men icke funktionella krav omfattar också alla andra krav som inte är funktionella krav.

b)

Kärnan i ett scenario eller ett användningsfall är att upptäcka funktionella krav genom att skriva historier av en användare som använder systemet för att uppnå sina mål.

Användningsfall: *Låsning av betalskåp (Primärt flöde)*

> *En nyttjare av skåpet anländer med en 10 krona. Skåpet är olåst, går ej att låsa och nyckeln går ej att avlägsna. Nyttjaren öppnar skåpet, lägger i saker och lägger 10 kronan i betalhålet på skåpdörrens baksida. Nyttjaren stänger skåpet, låser det, tar nyckeln och går iväg.*

Man skulle kunna säga att användarfall är de funktionella kraven på systemet samt krav på systemets beteende i en mer berättande form.

c)

Primärt händelseflöde är flödet där allt går som det skall och allt slutar lyckligt. Exempel på detta är *Låsning av betalskåp flödet* som hittas i uppgift 2b. Det kan bara finnas ett primärt flöde.

Alternativt händelseflöde är flödet där någonting händer på vägen men löser sig ändå, flödet beskriver hur systemet tar hand om det oväntade. Exempel på alternativflöden i betalskåpsscenariot kan vara att nyttjaren saknar 10 krona, men har en femkrona som fungerar och därav slutar lyckligt. Det kan finnas många alternativa flöden som fångar avvikelser ifrån primärflödet.

Exceptionella händelseflöden är flödet där någonting händer på vägen likt ett alternativt flöde, men löser sig inte till ett lyckligt slut. Exempel på detta i betalskåpscenariot skulle kunna vara att skåpet saknar nyckel och därav misslyckas nyttjaren.

Deluppgift 3 – Testning (300 ord)

a) (4p)

I en testplan bestäms vilket användningsfall och vilken testtyp tillämpas: I testtypen whitebox har utvecklaren tillgång till koden där största del av koden ska täckas med hjälp av testdata i motsvarighet till blackbox-typen där utvecklaren ser kodens behållare som en funktion som förväntas leverera ett värde. Efter detta så planeras lämplig testomgivning och anges i ett testplansdokument. Även bakgrund till testningen, referenser till användningsfall, definierade enhetstester med testansvarig och stoppregler, rutiner för dokumentation av testresultat finns i dokumentet. Innehållet varierar p.g.a. att mjukvaruutvecklingsprocessen kan sakna normativ mall för detta.

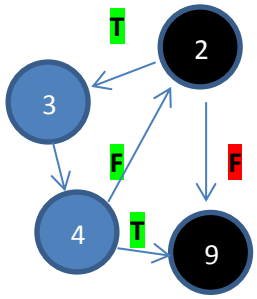
b)

En testsvit utgörs av flera testfall med tillhörande testfixturer (säkerställer att testresultaten är upprepbara). Exempelvis så utgör flera testfall som testar säkerheten hos klienten en testsvit för klientsäkerhet. Man bygger upp testsviter med fokus på olika delar i systemet. Testsviten innehåller också information om hur testmiljön konfigureras för testfallen.

c)

Kodtäckningsgrad (Whitebox testning) anger i procentantal hur stor del av koden som testats.

```
1 int countBetween (int i, int j) {
2    while (i < j + 10) {
3        i++;
4        if (i == j) {
5           break;
6        }
7
8    }
9    return i - 1;
10}
```



Funktionen ovan returnerar (ineffektivt) skillnaden emellan i och j.

Om man vill testa instruktionstäckningen i exemplet ovan med värdena i = 20, j = 30 så kommer vi att täcka in raderna 2, 3, 4 och 9. Det vill säga alla statements och får en täckningsgrad på 100% med ett enda testfall. Syftet med är att visa hur stor del av koden som testats av testfallen och används när man vill veta hur många instruktioner (statements) som exekverats. Man vill generellt uppnå hög täckningsgrad med så få testfall som möjligt.

Om man testar för grentäckning med samma värden så kommer vi att beröra alla True / False grenar med grön bakgrund. Vi missar alltså en möjlig väg. Med grentäckningen vill man visa vilka möjliga vägar i exempelvis if-satser som koden exekverats.

Dessa tester används för att visa hur stor del av koden som testats och visar ifall fler testfall behöver implementeras.

Deluppgift 4 – Planering (250 ord)

a)

I exempelvis Open UP så bryter man ner planeringen och arbetsuppgifterna i följande nivåer:

- Månader: fokus på projektens livscykel och kund. Här planeras projektet där det går igenom faserna Inception och Elaboration där riskerna elimineras för att sedan övergå till construktion och transition faserna där värde skapas för kund.
- Veckor: fokus på utvecklingsgruppen. Här planeras mål för iterationen och gruppen jobbar för att åstadkomma demonsterbara versioner av mjukvaran i slutet av iterationen.
- Dagar; fokus på personal. Här planeras dagliga arbetsuppgifter på den lägsta nivån för personal.

Projektets uppgifter bryts ned och planeras i mindre detaljer för varje nivå. Beroenden identifieras och tiden uppskattas, resursbehoven uppskattas för att allt sedan dokumenteras. Efter varje iteration granskas utfallet och en ny planering genomförs.

b)

I UP:s Inceptionfas (och Elaborationfas) ligger fokus riskarbete för att sedan övergå till att skapa värde. I de första iterationsfaserna uppmuntras målen vara till att identifiera och eliminera de största riskerna och att skapa visuella egenskaper som är viktiga för kunden.

Riskdriven (arkitekturcentrisk) iterativ utveckling förespråkar att man under de första iterationerna fokuserar på att bygga, testa och stabilisera systemets kärnarkitektur. Saknad av detta ses som en vanlig risk.

c) (3p)

I Sprint planeringen i SCRUM finns två steg.

Steg 1 handlar om att fastställa mål för sprinten med kundfocus. Man skapar en ”product backlog” (en lista med allt som ska göras) som skulle kunna innehålla uppgiften om att få igång mjukvaruutvecklingsmiljön och har hög prioritet. Utvecklingsteamet, Scrum Mastern och Product Owner deltar i mötet.

I steg två uteblir Product Owner där Scrum Mastern och Utvecklingsteamet bestämmer en ”sprint backlog”, en mer konkret form av lista med uppgifter som utvecklingstemat ska lösa. Exempelvis att en databasserver ska installeras. Utvecklingsteamet bestämmer själva i båda stegen.

Deluppgift 5 – Agila processer (250 ord)

a) (6p)

Varje sprint är ett iterativt inkrement. I slutet av en sprint ska ett körbart demo producerats, denna utökas genom varje sprint.

Första delen av sprinten består av Sprint planning. Scrum mastern, Product Owner och utvecklingsteamet deltar i planering i steg 1, Product Owner uteblir i steg 2.

Daily scrum är ett kort dagligt möte (15 minuter). Utvecklingsteamet och Scrum mastern utvärderar arbetet och planerar resten av sprinten för att nå målen i product backlog. Under Daily scrum presenterar teammedlemmar vad denne; gjort sedan igår, ska åstadkomma tills imorgon, eventuella hinder. Mötet omfattar inte problemlösning eller för att se vem som ligger efter utan medlemmars åtaganden och ge Scrum mastern en överblick.

I slutskedet sker Sprint Review där återkoppling till Product Owner sker. Vad som åstadkommits under sprinten redovisas i features och körbar demo. Här medverkar även utvecklingsteamet, Scrum mastern, kunder och utvecklare från andra projekt.

Sist sker Sprint Retrospective där teamet reflekterar över förbättringar under nästa sprint. Även Product Owner och Scrum master deltar.

b)

Parprogrammering:

Eftersom det är svårt för ensamma utvecklare att hålla koll på både bra syntax och hur modulen kommer att passa i produkthelheten så finns parprogrammering. Där fokuserar en på korrekt kod, en annan på att koden passar in i produkthelheten. Automatisk kunskapsdelning uppnås då paret får insyn i samma arbete.

Test Driven development:

Detta innebär att enhetstestkoden skrivs före själva programkoden som sedan testas mot enhetstestkoden vid utveckling. Testkoden skrivs före programkoden med en idé om hur den ska se ut. Alla enhetstester skrivs inte på en gång, utan detta görs i mindre steg som sedan testas aktuell programkod. Efter programkoden validerar emot testerna fortsätter processen.

Fördelar:

- Testkoden blir skriven.
- Automatiskt planeraring av klasser.
- Ger självförtroende för kodändringar då tester enkelt indikerar fel.