Systemutvikling

Obligatorisk oppgave 3

GRUPPE 27
Adrian Westlund – s198571
Hallbjørn Storruste – s165519

Oppgave 1: Prosjektplanlegging

a)

A: Kravarbeid
B: Utforming av system og design
C: Implementering
D: Enhetstesting
E: Integrering
F: Systemtesting
G: Validering og verifisering
H: Innføring
I: Vedlikehold

b)

Aktivitet	Avhengighet	Tidsforbruk(uker)	Milepæl
A	Ingen	4	
B	A	2	M1
C	B		M2
D	B	8	
E	C, D	2	M3
F	C, D	2	
G	E, F	2	M4
H	G	2	M5
I	G	2	

c)

	2	4	6	8	10	12	14	16	18	20
A	Ingen									
B			A							
C				B						
D				B						
E								C, D		
F								C, D		
G									E, F	
H										G
I										G

d)

Risiko	Sannsynlighet	Konsekvens	Tiltak	Ansvarlig
Finansielle problemer hos kunden	Lav	Katastrofal	Frigjøre/anskaffe midler.	Kunde
Umulig å anskaffe kompetent arbeidskraft	Høy	Katastrofal	Midlertidig ansettelser/ outsourcing. Kursing av personalet.	Prosjektleder
Nøkkelpersoner i prosjektet blir syke.	Moderat	Alvorlig	Høy bemanning	Prosjektleder

Databasen brukt i systemet takler ikke så mange transaksjoner som forventet.	Moderat	Alvorlig	Ta ekstra høyde for mange transaksjoner ved valg av database.	Prosjektleder
Forandringer i kravspesifikasjonen fører til store forandringer i systemet.	Moderat	Alvorlig	Grundig arbeid med krav/ brukerhistorier. Jevnlige møter med kunden.	Kunde/ Prosjektleder / kundekontakt

Oppgave 2: Estimering

a)

Vi antar att vi har lite erfaring från liknande proskekt och har lagt ner mycket tid på en god kravspecifikation. Med en god kravspec vill estimerigen bli mer precis och ge oss ett avgrensat område (scope) och det medför mindre risk för att prosektets omfang vill utökas.

Vi har sätt på de tre olika estimerings tekniker, LOC, FP och COCOMO och har bestämt oss för att använda FP (Funkstionspoäng). Vi har valt att använda funktionspoäng då vi tror att det är lättare att estimera antal input, output mm, då dessa är något som vi kommer att veta utifrån vår kravspec. Med lite eller ingen erfaring från liknande prosjekt blir det svårt att komma fram till ett relevant antall kodelinjer.

b)

Det finns liknande systemer från förr och detta gör kompleksiteten mindre eftersom antagelserna blir mer precisa. Detta medför att risken för att kravspecifikationen skal ändras minskar. Storleken på systemet bedömer vi till att vara medelstort och storleken har stor betydelse på kompeksiteten.

Oppgave 3: Arkitektur

a) Den fysiska arkitekturen ger en överikt över vilka enheter och maskiner som systemet använder. Medans den logiska arkitekturen är en översikt vilka komponenter som är i programvaran.

b) Trelags logisk arkitektur.

Lag 1: Brukargränsnitt
Lag 2: Business-logikk lag
Lag 3: Datalag

c) Firelags fysisk arkitektur.

Lag 1: Klient (Mobil)
Lag 2: Mobiltjener
Lag 3: Applikasjonserver
Lag 4: Database

d)

Vid lagdelt arkitektur kan ett lag i princip bytas ut utan att resten av lagen må ändras.

e)

Klient-server: Systemets funktioner är organiserat i tjänster och de skickas från en egen server. Detta ängnar sig bra när det är många som ska ha tillgång till samma ställe. Man kan vid behov spegla en server för att utöka kapasiteten vid behov.

MVC: Struktureras i tre komponenter som komuniserar med varanadra.
Model: Hanterar data och hur datan ska behandlas.
View: Bestämmer och hanterar hur datan skal visas för användaren.
Controller: Hanterar val av användaren som musklick mm och skickar intstruktioner till View och Model.

SOA: Använder tjänster som inte är beroende av programvaran. Fokuset ligger på lösa kopplingar som är lätta att ersätta eftersom tjänsterna inte är kopplade mot varandra. Den använder istället förhands definerte prorokoller och stanader för att skicka besked till varandra. Tjänsten är även inte språkberoende.

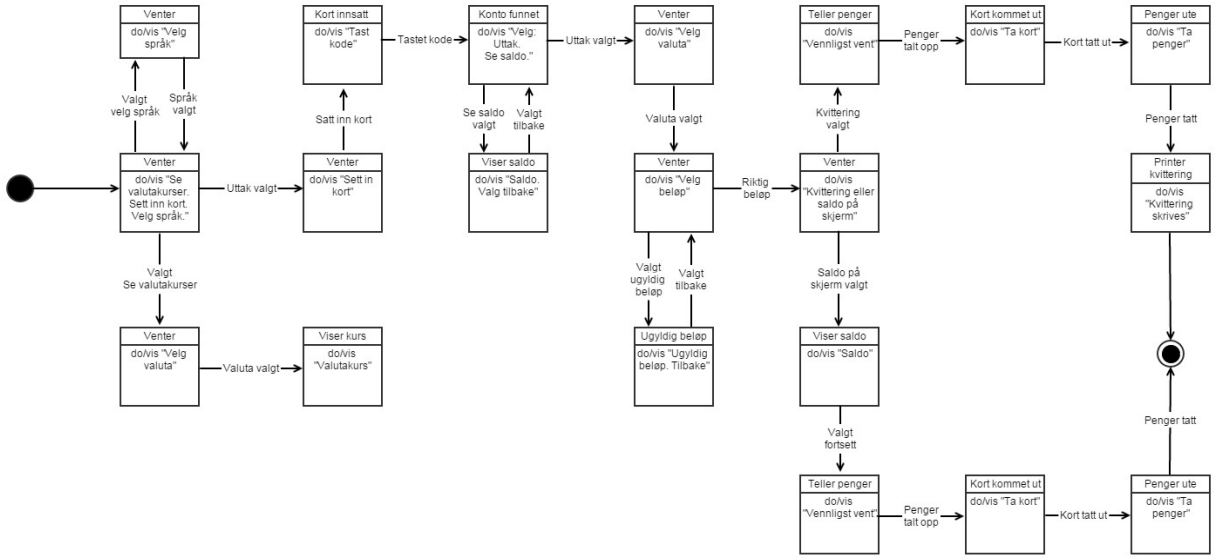
f)

Lagdelt arkitektur kan göra systmet lånngsamare då en tjänstförfrågning måste tolkas av flera lag.

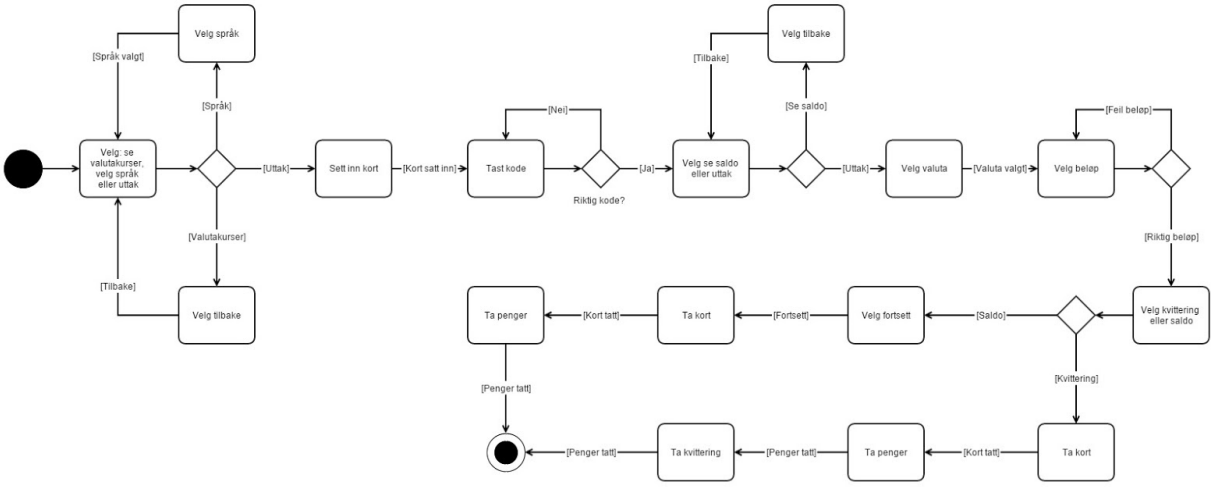
Oppgave 4: Tilstand- og aktivitetsdiagrammer

a) Minibanksystem: Tilstandsdiagram

Man kan avbryte fra alle tilstander.



b)Minibanksystem: Aktivitetsdiagram
Man kan avbryte fra alle aktiviteter.



c) De begynner med en sort prikk og avsluttes med en sort prikk med hvit ring rundt.

d) Et tilstandsdiagram viser hvordan systemet endrer sin tilstand fra etter de ulike hendelsene. Dette er nyttig for å finne deadlocks i systemet. Dvs hvor flyten går i en loop eller ender i en tilstand uten å kunne komme videre eller ut av den. Den viser hvilke valg som får oss fra en tilstand til en annen. Som i embeddede systemer som en

mikrobølgeovn, hvor man må gjøre noen valg og ha sikkerhet med at døren skal være lukket før programmet kan settes igang.

e) Et aktivitetsdiagram viser hvordan systemet beveger seg fra aktivitet til aktivitet ettersom hver aktivitet utføres. Dette kan være nyttig for å finne deadlocks i et system. Det er også nyttig for å forstå arbeidsflyten i systemet og se hvilke aktiviteter som kan kjøres parallelt. For eksempel for en kaffemaskin: kaffen kan kvernes mens vannet varmes.

Oppgave 5: Testing

a)

Enhetstesting är en individuell test som körs på en del av programmet, Denna test görs av utvecklaren och den används främst för testing av objekter, klasser och metoders funktionalitet

Integrasjonstesting är när man testar flera delprogram samtidigt och målet är att se om programmet fungerar som det ska även när det är sammasatt. Även detta utförs av utvecklaren.
Här används det fyra typer av test, Big bang, Bottom-up, Top-down och Sandwich.

Systemtesting innebär att man testar hela systemet och man testar även systemet tillsammans med andra systemet så som hardware. Detta utför utvecklaren och man testar även om systemet möter alla krav som är satta.

Akseptansetesting innebär att hela system skal testas oss klienten och det finns flera tester som används. De olika är tex smoke test, ytelsetest, vedlikeholdtest mm.
Det görs även alphatest som görs av utvecklaren och senare betatest, Betatesten testas av “riktiga” användare.
Målet är att kunna bekräfta att system är det kunden vill ha och att det uppfyller kraven till kunden.

b)

De olika testerna används i olika delar av processen Enhetstesting är något man snabbt kan börja med och det gör kontinuerligt genom hela utvecklingsfasen.
Allt efter som systemet blir större kan man börja köra de olika delarna mot varandra och då använder man integrasjonstetsting. Här vill man testa och det inte uppstår fel när de olika delarna i programmet “pratar” med varanndra. Dessa två tester kan man köra parallellt.
När alla delar i systemet är klara kan man börja med systemtestning.

När dessa tester är klara så ska inte utvecklaren testa mer utan man ska ta in utomstående som inte kan systemet och vet hur det är uppbygt och då använder man sig av akseptansetesting.

c)
Sjekke billettstatus: Liten test och är inte beroende av andra delar av systemet och kan därför testes med en Enhetstesting.
Innsjekking: Här måste delar av programet “prata” med varandra och därför måste vi använda oss av Inegrasjonstesting.
Sjekke pratiskt info: I denna testen måste vi använda interna och externa systemer för att få ut riktig info. För att kunna tolka denna infon måste den bli testad av användarna som sitter på kompitansen att avgöra om det är riktig eller felaktig info och därför måste vi här göra en akseptansetesting.
Annsatte og VIP: För att kunna hämta datan som krävs måste vi använda hela det interna systemet och det gör vi med en systemtestning.