SUM 4. semester projekt

Design Classique – Salgsmodul

Christopher Buch, Lasse Lund, Lars S. Jensen og Tobias Jensen.

6-6-2014

Indhold

Indledning (Christopher)

Til dette projekt har vi valgt at udvikle et færdigt salgssystem, til firmaet Design Classique(DC). DC har ønsket sig en færdig løsning, til hjælp med salg af varer på en webfront. Til dette har vi tænkt os at udvikle et salgsmodul til et ERP system, som firmaet kan bruge internt, samt en webshop hjemmeside, hvorfra kunder selv kan vælge at foretage køb. Som en del af dette projekt, vil vi benytte systemudviklingsværktøjer, til at give os et indblik i hvordan vi bedst kan opfylde kundens behov. For at give kunden et overblik over opgavens omfang, foretager vi os en estimering af hvor lang tid, et projekt som dette vil tage, hvor meget det vil koste og hvilke risikoer et projekt af dette omfang vil have. Der vil efter et grundforløb, hvor projektgruppen udspecificerer de grundlæggende dele af systemet, blive arbejdet med den agile proces SCRUM. Dette gøres for at sikre at kunden løbende kan følge med i hvor langt projektgruppen er nået med systemet, samt tillade dem at komme med ændringer der optimerer systemet netop til deres formål.

Kravspecifikation(Alle)

Webshoppen

Til firmaet design classique, skal laves en webshop, til fremvisning af firmaets varer, samt hvorfra kunder kan købe disse varer.

Hjemmesiden vil blive opbygget af html5, med php, javascript og CSS.

Der vil til hjemmesiden skulle være brugere. Hertil vil der være en administrator(salgschef) og kunder.

Kunder oprettes med navn, adresse, by, telefon nummer og gyldig email.

Det vil sige email skal verificeres når kunder opretter sig.

Kunder skal være i stand til at foretage CRUD på deres egen bruger.

Hvis registrerende kunder fortager et køb på hjemmesiden, skal kunden information automatisk vises ved næste køb.

Systemet vil ikke gemme kunders konto eller kort informationer (dette grundet sikkerhedskrav).

Salgschefen kan foretage CRUD operations på varer, dette også i tilfælde af at der er sket prisfejl på en vare, faktura hvis bestillinger ændres og kunder hvis kunden skulle få nye oplysninger.

Medarbejdere kan også fortage CRUD på kunders ordre, samt kunde oplysninger.

ERP - systemet

Til webshoppen skal laves et ERP system, således at firmaet kan holde styr på indkøb, salg og lagerstyring. Dette vil blive koblet til en database, som også skal indeholde kundeoplysninger, så kunder kan registreres og der kan foretages statistik over deres køb.

I databasen skal det være muligt at se betalingsstatus for kundens fakturaer, således at der kan ses om en kunde har betalt for varer bestilt.

I databasen skal det desuden være muligt at se betalingsstatus for ordrer, som virksomheden har bestilt.

Databasetypen der vil blive brugt til dette er en MySQL database.

Det skal være muligt for salgschefen at udføre CRUD på medarbejdere, varer og ordre i systemet. Samtidig skal medarbejdere også kunne udføre CRUD på fakturaer for kunder.

I forbindelse med lagerstyring skal systemet automatisk opdatere ved salg af varer og hvis denne vare kommer under et hvis antal, skal der sendes besked om mangel på denne vare.

Ved hjemkomst af varer tastes disse manuelt ind i systemet.

I systemet skal det være muligt at udskrive en liste over lagerstatus fordelt på de forskellige varegrupper.

Da virksomheden køber brugte varer hos kunder skelnes der mellem nye og brugte varer, som har forskellige id. Samtidig skal det være muligt at krydse af om brugte varer er under renovering.

---PERFORMANCE---

Det må i ERP systemet ikke tage mere end 30 sekunder at skulle oprette en vare.

Det skal for en kunde ikke tage mere end 2 minutter, at oprette sig som kunde på hjemmesiden.

Fra at ERP systemet bliver opdateret, til at webshoppen bliver opdateret skal være atomar.

Projektledelse (Tobias)

Til udarbejdelse af afsnittet om projektledelse har vi brugt Software Engineering, A Practioner's Approach 7. edition, kapitel 5 og 24

Gruppekontrakt

Fælles kommunikationsmiddel: Facebook gruppe. Udover kommunikerer vi via sms/telefon og på Skype. Vi har oprettet en telefonkæde, som benyttes hvis man mener, at man kommer for sent.

Arbejdstid aftales på daglig basis. Idet at opgaveperioden indeholder en del ferie/helligdage har vi besluttet at aftale arbejdsdage på skolen hver gang vi mødes. Desuden tilstræbes det at holde en slags møde (eventuelt SCRUM møde) hver dag der er afsat til SUM, om det så er på skolen eller via Skype.

I tilfælde af sygdom melder pågældende medlem sig syg hurtigst muligt.

Værktøjer til udarbejdelse af projektet
Tekstbehandlingsværktøj: Microsoft Word
Skærmbilleder/GUI: Visual Studio
Diagrammer: Dia / Visual Studio

Projektet

The People

Vores projektgruppe er relativt lille med en størrelse på blot 4 mand, og estimeringen af projektet er derfor lavet ud fra disse forudsætninger. Vi har samtidig taget vores forholdsregler og budgetteret projektet med, at vi skal have en konsulent ind over, hvis/når vores kompetencer ikke er nok til at opfylde kundens behov.

The Product

Vores indledende arbejde bestod hovedsageligt af kontakt med kunden, Design Classique, som valgte at kontakte os da de havde brug for en IT løsning. I forbindelse med udvidelse af Design Classique's forretningsområde, har de besluttet at tilføje nye midler til forretningen. Det har de gjort med et nyt lager, som ligger andetsteds end deres butik.

I forbindelse med udvidelsen har de indset, at de får brug for yderligere eksponering. Det skal ske med en webshop, som kan bruges til at fremvisning og salg af varer, samt yderlige services som renovering af møbler. Webshoppen skal kobles op med ERP systemet, så man på webshoppen kan se om en given vare er på lager.

Efter yderligere kontakt med kunden afleverede de en udspecificeret kravspecifikation, og ud fra kravspecifikationen, har vi opsat nogle problemer i hovedpunkter, som skal løses for kunden:

ERP system
- Mangelfuld lagerstyring
- Lette salg og fakturering
- Skabe gennemsigtighed ved salg af varer

Webshop
- Eksponering ved hjælp af webshoppen
- Styrke salg med webshoppen

I øjeblikket er det besværligt at bogføre hvilke varer virksomheden får hjem og hvilke varer der findes på lageret. For at kunne holde styr på det samlede lager ønsker de et salgsmodul til ERP-systemet, så de hele tiden ved hvad der er på lager. ERP-systemets nye salgsmodul skal samtidig gøre det nemt at se hvem der har solgt hvilke varer til hvilke kunder.

Derudover er det selvfølgelig estimeret, at kunden på senere tidspunkt kan kræve flere moduler / features til systemet, men som det ser ud nu er kravene som ovenstående. For at skabe de bedste rammer for produktet har vi taget vores forholdsregler og sammen defineret en kravspecifikation ud fra kundens krav.

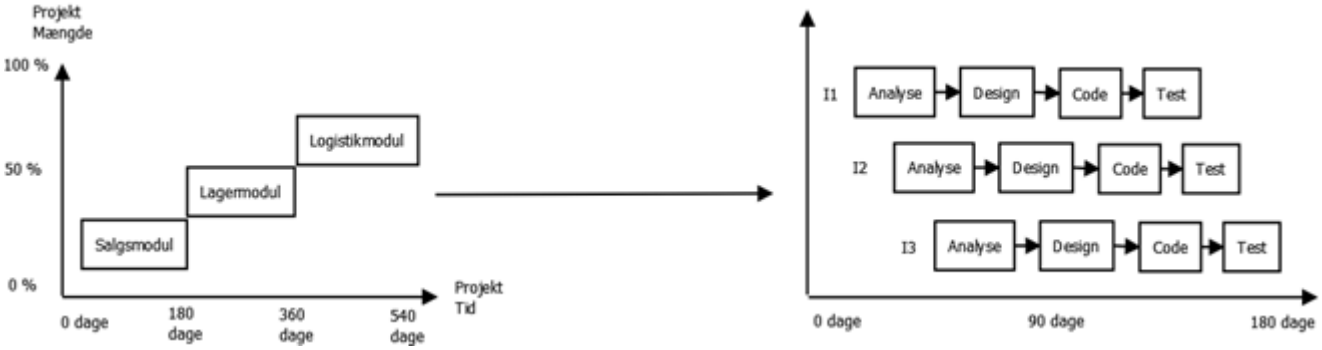
Kunden havde i sit første udkast angivet et ønske om funktionalitet, der kunne implementeres på et iOS system. Fra personalets iPads, iPhones og MACs skulle det være muligt at læse fakturaer, ringe til kunder, finde kunder via GPS og tilføje billeder til varer fra kamerarullen.

Da vi følte det var uden for vores kompetence område afviste vi hurtigt, at dette var muligt at gennemføre indenfor den givne timeboks og betalingsramme. Risikoen ved at give os i kast med denne del af systemet i forhold til udbyttet var for stort.

The Process and The Project

Til vores projekt har vi benyttet os af den inkrementelle model. I den inkrementelle model udvikler man mindre dele af systemet ad gangen hvorefter man gentager cyklussen med næste inkrement. Således er modellen en iterativ model, som hele tiden skifter mellem udvikling af analyse, design, kode og test.

Vores projektperiode har kun omfattet analyse og design af salgsmodulet i første inkrement, hvor tanken var at udføre hele første inkrement, som en skabelon for resten af projektet som omfattede salgsmodulet. Vi har taget udgangspunkt i vandfaldsmodellen, hvor vi har beskæftiget os med fordybelse i analyse og design, fordi vi ønskede at have et godt fundament for resten af arbejdet med salgsmodulet. Vi har dog kun beskæftiget os med cirka 10 % af use cases og skabeloner for klassediagram samt arkitektur, sekvensdiagram og funktionsanalyse. Resten af første inkrement vil fortsat være at udvikle en skabelon for kodningen i de følgende inkrementer. Denne fremgangsmåde medfører en høj grad af completeness, men en risiko for, at vi har misforstået systemet.



Projektet er selvfølgelig ikke fastlåst til 3 inkrementer. Dette er blot en illustration af den inkrementelle model.

Problemstillingen er, at vi ikke ved om det vi har designet kommer til at fungere i praksis for kunden, da vi endnu ikke har præsenteret noget for kunden. Først når vi mødes med kunden, og diskuterer det designede system, ved vi om det var præcis hvad kunden ønskede/havde behov for. Viser det sig at vi har taget fejl kommer det til at koste os dyrt.

Modsat så kompenserer vi for denne ”fejltagelse” ved at afvikle de følgende inkrementer som SCRUM. Der er dog stadig en chance for at vores ”fejltagelse” viser sig som en fordel, hvis vi alligevel har ramt hvad kunden har tænkt systemet skal kunne.

De efterfølgende inkrementer omhandlende salgsmodulet har vi, som omtalt, planlagt at udføre agilt med brug af SCRUM. Derfor har vi estimeret resten af projektet i forhold til SCRUM sprints på 14 dage. Disse dele vil ikke kun omfatte kodning og test, men videreudvikling af use cases, klassediagram og brugerflader. Når salgsmodulet er færdigudviklet kan man starte forfra med første inkrement for næste modul i ERP-systemet, eksempelvis et lagermodul.

Software projekter kræver, at man opfylder kundens behov, ikke overskrider ressource grænser og samtidig overholder deadline. Øger man scope kan det komme til at koste flere penge og tage længere tid, mindre tidshorisont betyder mindre scope og flere penge, mens et lille budget mindre scope men mere tid. Der er

dog ingen garanti eller regelsæt, der siger at man skal holde sig indenfor trekantens grænser. Typisk vil man i et projekt fastholde et parameter, mens de andre to skal kunne varieres, hvilket skaber en vis usikkerhed i projektet fra start.

En agil proces model er oftest mest villig til at bibeholde tiden, mens den kan ændre på scope og cost. Kan vi få flere folk på projektet så vi kan blive færdige til tiden eller skal vi skære ikke-væsentlige dele af projektet? At man overholder tiden har et begreb: Timeboxing. Vi ved altså helt præcis hvad tidsrammen for projektet er og når vi er ved deadline, så skal projektet være færdigt. Samtidig gør timeboxing også op med perfektionisme, som godt kan være et problem i IT verdenen.

Modsat er den traditionelle vandfaldsmodel fokuseret på at opnå projektets krav til fulde og mere interesseret i at ændre på tiden projektet kommer til at tage og hvor meget det kommer til at koste.

Det er vigtigt at kunne mestre balancegangen indenfor projekttrekanten for at opnå et helt igennem succesfuldt projekt. Sandheden er dog at langt de fleste projekter ikke udelukkende ender som succeser, men enten som succesful failure eller failed succes.

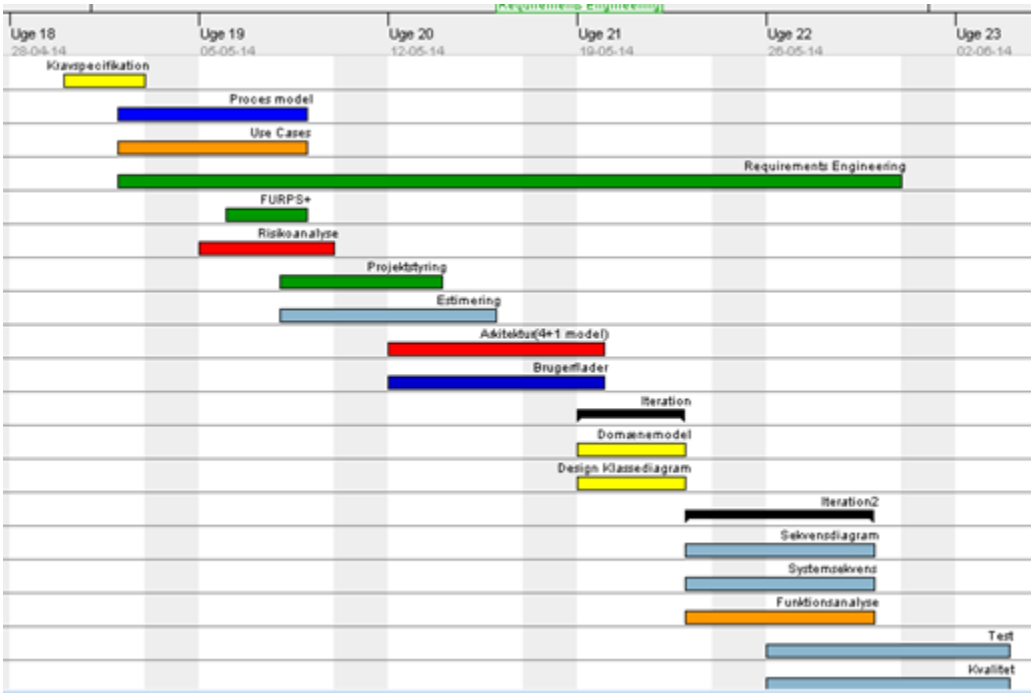
Failed succes er, når man afleverer projektet til kunden til tiden, men med fejl og mangler. Positivt er at man afleverer til tiden, men er projektet langt fra færdigt kan kunden vælge at afbryde samarbejdet.

Succesful failure er, når man afleverer projektet efter den aftalte deadline, men hvor projektet indeholder de features, som kunden har bedt om. Fordele ved dette er at kunden får det han har bedt om, men det er selvfølgelig ikke altid man lige kan få lov til at overskride deadline. Kunden kan forlange, at man bliver ved med at supportere, vedligeholde og opdatere projektet, som tilbagebetaling for den overskredne deadline.

Projektgruppen kan hurtigt blive enig om, at ender man i en situation hvor vi kan se at vi ikke når deadline, så vil vi forlænge projektet og færdiggøre modulet. Der er ingen garantier for at estimeringen af projektet holder, men modsat skal der også tages højde for lignende situationer under risikostyring.

Tidsplan

Foreløbig tidsplan under uændrede ressourcer



Den foreløbige projektplan blev udarbejdet før valget af procesmodel skete. Tanken i starten af projektet var at lægge vægt på gennemarbejdelse af indledende øvelser før projektets start. Samtidig har vi taget udgangspunkt i en et projekt med iterationer, som først for alvor begynder når rammerne for projektet er defineret.

Den endelige tidsplan

Den endelige tidsplan blev selvfølgelig tilpasset efter vi havde fundet ud af hvilken procesmodel der var relevant for vores projekt. Planen er blevet delt nogenlunde op efter farvekode, så det er muligt at se hvem der har lavet / skulle lave hvad. Farvekode er som følger:

Lasse = Blå Christopher = Orange Lars = Rød Tobias = Grøn Alle = Gul

Efter at have fastslået at tage udgangspunkt i den inkrementelle model med brug af SCRUM kunne vi planlægge projektets videre forløb. Vi arbejdede os stille og roligt ind i projektet og fik overblik over hvad der var nødvendigt at udarbejdet i første inkrement. Desværre nåede vi kun cirka halvdelen, analyse og design. Projektet tog en uventet drejning og vi blev nødt til at sætte det på standby i cirka en uge for at kunne lave andre vigtigere opgaver.


GANTT projekt
Navn
Start dato
Slut dato
Kravspecifikation 30-04-14 02-05-14
Proces model 02-05-14 08-05-14
Use Cases 02-05-14 08-05-14
Requirements Engin... 02-05-14 13-05-14
Risikoanalyse 05-05-14 13-05-14
FURPS+ 08-05-14 08-05-14
Estimering 09-05-14 09-05-14
Domænemodel 14-05-14 14-05-14
Estimering 14-05-14 20-05-14
Arkitektur 15-05-14 21-05-14
Klassediagram + Ark... 16-05-14 21-05-14
Funktionsanalyse 19-05-14 21-05-14
EER model 20-05-14 21-05-14
Kvalitet 28-05-14 30-05-14
Funktionsanalyse 28-05-14 02-06-14
Systemsekvensdiagr... 28-05-14 03-06-14
Brugerflader + Usabi... 28-05-14 03-06-14
Sekvensdiagram 28-05-14 03-06-14
Klassediagram + Ark... 28-05-14 04-06-14
Arkitektur 02-06-14 05-06-14
Test 03-06-14 05-06-14
Uge 18
Uge 19
Uge 20
Uge 21
Uge 22
Uge 23
Kravspecifikation
Proces model
Use Cases
Requirements Engineering + projektledelse
Risikoanalyse
FURPS+
Estimering
Domænemodel
Estimering
Arkitektur
Klassediagram + Arkitektur
Funktionsanalyse
EER model
Kvalitet
Funktionsanalyse
Systemsekvensdiagram
Brugerflader + Usability
Sekvensdiagram
Klassediagram + Arkitektur
Arkitektur
Test

Valg af procesmodel(Lasse)

Vi har valgt at bruge Todd Little "Context - Adaptive Agility: Managing Complexity and Uncertainty" til, at vælge hvilken procesmodel vi skal bruge.

Så vi starter med at vælge, hvilken score vi skal have i de forskellige attributter.

Vi har taget udgangspunkt i vores gruppe og hvad vores kompetencer er.

Complexity

Attribute	Answer	Score
Team size	**4**	**3**

Vi er 4 i vores gruppe.

Mission criticality	**Small user base**	**3**

Hvis dette projekt ikke lykkes, vil det kun være den tid og penge som virksomheden vil miste, da det er et nyt ERP system og webside vi skal lave til Design Classique.

Team location	**Same room**	**1**

Da vi er en gruppe der ikke bor i nærheden af hinanden, har vi valgt at arbejde på skolen for, at vi har en bedre kommunikation med hinanden.

Team capacity	**New team of mostly novices**	**10**

Ingen i gruppen har før arbejdet med, at programmere og opsætte et ERP - system i en virksomhed.

Domain knowledge	**Developers require some domain assistance**	**5**

Vi har arbejdet med små ERP - systemer før, så vi har brug for noget hjælp til hvordan det skal programmeres.

Dependencies	**Moderate**	**5**

ERP - systemet og websiden er meget afhængige af hinanden. Hvis en vare bliver solgt på hjemmesiden skal ERP - systemets lager opdateres eller hvis en vare kommer på lager igen, skal hjemmesiden opdateres med det samme.

Uncertainty

Attribute	Answer	Score
Market uncertainty	**Minor changes in the market target expected**	**3**

Design Classique handler med møbler til en snæver målgruppe, der ligger blandt overklassen. Det vil sige, at deres målgruppe ikke ændre sig særlig meget og der vil ikke ske de store ændringer på 6 måneder.

Technical uncertainty	**Some incremental research involved**	**7**

Vores team er nye på området og derfor skal vi nok lave en del søgning, hvordan vi laver det bedste resultat.

Project duration	**12 months**	**5**

Da vores team på 4 personer ikke har lavet en projekt i dette omfang før, forventer vi at det vil tage 12 måneder.

Dependencies, scope flexibility	**Scope have some flexibility**	**5**

Teamet mangler kompetencer og erfaring inden for scope, hvilken begrænser fleksibilitet og scope.
Det giver tilsammen en score i Complexity på 7,34 og uncertainty på 6.56.
Hvis vi sætte disse tal ind i vores Houston Matrix quadrant assessment, som findes i vores bilag kan vi se at, vi befinder os i Colts, hvor 2 procesmodeller vil være det optimale valg XP eller scrum.
Vi har valgt, at arbejde med vandfald og scrum, da den vil give vores team et bedre overblik over hvor langt vi er ved hjælp af vores release backlog, sprints og burndown chart efter 1 inkrement.
Så efter vores 1. inkrement starter vi op på vores scrum sprints.

Der vil også være et 15 minutters møde hver dag, hvor ejeren, scrum masteren og resten af teamet sidder sammen og holder styr på at processen går den rigtige vej.
Grunden til vi ikke har valgt XP er, fordi vi mener at der er for lidt dokumentation i de forskellige faser i XP.

Risikoanalyse (Lars)

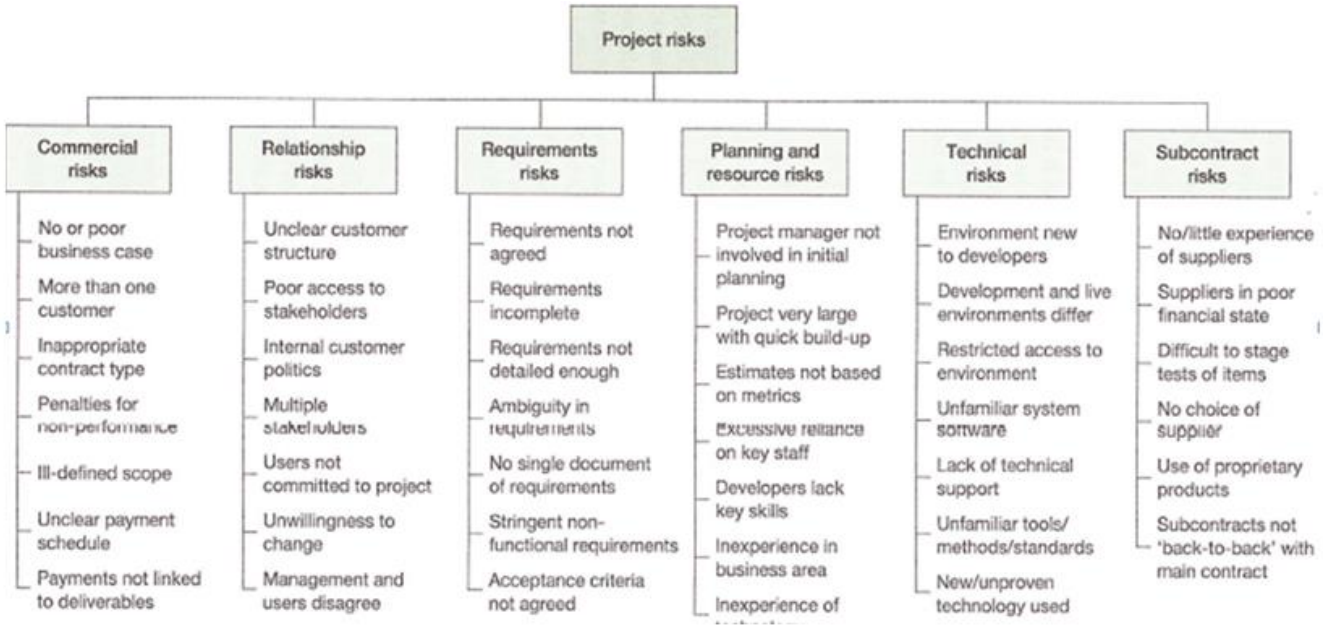
Til udarbejdelse af vores risikoanalyse, har vi anvendt metoder samt begreber fra bogen Software Engineering A Practitioner's Approach, 7ed. Kapitel 28.

Reaktiv eller proaktiv strategi?

En proaktiv strategi starter længe før, det tekniske arbejde igangsættes. De potentielle risici nedskrives samt deres sandsynlighed og konsekvens. Efter nedskrivningen vil man herefter etablere en risikostyringsplan. En reaktiv strategi betyder, at man ikke forbereder sig på mulige problemer, men først tager højde for dem, når/hvis de opstår. En sådan strategi falder ikke i vores interesse, idet vi så vidt muligt foretrækker, at være forberedte på de konsekvenser, der kunne opstå i projektets varighed. Dette passer også godt med at vi i første inkrement har fokus på de højest prioriterede analyser. At udarbejde risikoanalyse ligger i analyseudviklingen. Vi har, i gruppen, derfor valgt at anvende en proaktiv strategi, da vi mener at det er den bedste strategi for vores udarbejdelse af projektet. For projektet har vi valgt at arbejde iterativt,(dog ikke i vores første inkrement) hvilket har den betydning for risikostyringen, at vi vender tilbage samt holder opfølgning på de risici, som vi måtte have fundet frem til. Opfølgningen vil bestå i at tilføje nye risici, hvis vi støder på nogle, samt fjerne risici som ikke længere er relevante.

Risikokategorier

I vores risikotabel, som ses under afsnittet 'Risikotabel', har vi ud for hver risiko opsat en kategori. Kategorierne, som vi har fået inspiration fra, kommer fra nedenstående figur. Figuren er fra 'Cadle & Yeates. Project Management for Information Systems. Pearson. Prentice Hall 2008'.



Kategorierne i risikotabellen opdeles forkortelserne, som ses her:
Commercial risk – CR, Relationship risk – RR, Requirments risk – RQR, Planning and resource risk – PR, Technical risk – TR, Subcontract risk – SR

Risikoestimering

Vi har anvendt figur Lars1 (ses i bilaget) til at estimere konsekvenserne for de risici , som vi har fundet frem til. De risici, som vi har fundet frem til, ses i næste afsnit.

Ud fra figuren har vi opsat nogle tal under 'konsekvens' for at vurdere, hvilken konsekvens risikoen kan have for projektet, hvis konsekvensen skulle indtræffe. Ud fra figuren har vi vægtet tallene på følgende måde:

- Katastrofe vægtes som 1
- Kritisk vægtes som 2
- Marginal vægtes som 3
- (Negligible) Ubetydelig vægtes som 4.

Risikotabel

Herunder ses vores risikotabel, hvor der er blevet indsat en antaget sandsynlighed og konsekvens. Tallene under konsekvens er blevet vægtet ud fra figur 28.1 fra bogen 'Software Engineering A Practitioner's Approach, 7ed'.

Risiko	Kategori	Sandsynlighed	Konsekvens	RMMM
Kunden ændrer krav til systemet.	RQR	80 %	3	ID_01
Projektet er mere komplekst end antaget.	PR	30 %	2	ID_02
Manglende systemdata grundet backup failure. Dette kan skyldes manglende opfølgning på backup eller at programmet, der anvendes som backup, svigter.	TR	10 %	1	ID_03
System opfylder ikke kundens behov.	BU	10 %	1	ID_04
Udviklerne har manglende erfaring med programmer i møbelbranchen.	PR	80 %	4	Ikke relevant
Udviklere bliver ramt af sygdom, hvilket resulterer i at udviklingsprocessen bliver langsommere.	PR	20 %	3	Ikke relevant
Udviklerne mangler kompetencer i værktøjerne/teknologierne.	TR	15 %	2	Ikke relevant
Urealistiske forventninger til systemet.	PR	10 %	2	Ikke relevant
Ny/ikke afprøvet teknologi anvendes.	TR	5 %	2	Ikke relevant
Kunde vælger en anden løsning til systemet, hvilket medfører en ny leverandør. Vores opgave afbrydes.	BU	5 %	1	Ikke relevant

The Cutoff Line: I bilaget kan ses figur Lars2, hvor man sorterer de irrelevante risici fra. Figuren går i alt sin enkelthed ud på, at man kigger på en risiko og vurderer om det er nødvendig at lave et RMMM ud af denne risiko(forklares senere). I Software Engineering A Practitioner's Approach, 7ed. Kapitel 28 står der følgende: "A risk factor that has a high impact but a very low probability of occurrence should not absorb a significant amount of management time. However, high-impact risks with moderate to high probability and low-impact risks with high probability should be carried forward into the risk analysis steps that follow. All risks that lie above the cutoff line should be managed." Altså fokuserer vi kun på de risici, som er over den sort linje i tabellen. Den sorte linje er cutoff linien.

Assessing Risk Impact

Vi har i vores projekt valgt at lave en risikoestimering for vores risici, der ligger over "the cutoff line". Det skal nævnes, at vi kun har regnet prisen ud for de faser, hvor vi programmerer og viser hvad vi har lavet for kunden. Analyser af eksempelvis klassediagrammer med mere har vi ikke regnet på, da vi mener at vi kan nå dette, og gøre det tilfredsstillende for kunden. Risk exposeren udregnes på følgende måde: RE = P * C hvor C står for hvad det vil koste projektet, skulle risikoen indtræffe og hvor P står for sandsynligheden for at risikoen indtræffer. Vi har valgt at vise estimeringen for to af de nævnte risici, i denne rapport. De øvrige estimeringer findes i bilaget!

Risk Identification: Kunden ændrer krav til systemet.
Risk Probability: 80 %
Risk impact: Vi har valgt at vægte denne risiko 3 under konsekvens. Havde vi valgt at benytte udelukkende en vandfaldsmodel under vores projektudvikling, ville konsekvensen muligvis have været større(Hvilket havde givet en mindre vægt/tal under konsekvens i risiko tabellen). Det skyldes, at vi først ville præsentere det udviklede system for kunden til sidst i vores udviklingsproces, hvilket ville medføre en højere sandsynlighed for, at kunden ville være utilfreds med systemet. Vi har valgt en inkrementel procesmodel, hvilket betyder kunden har mulighed for at sætte sit præg på systemet i iterationerne, som kommer efter vores første inkrement. Med dette menes der, at der vil være fremvisning af det system vi udvikler for kunden flere gange under projektets levetid. Grundet vores valg af en inkrementel model er konsekvensen derfor ikke så høj, idet vi finder ændringer nogenlunde tidligt i udviklingsfaserne sammen med kunden. Grunden, til at den ikke er vægtet 5, men 3, er fordi ændringer ofte kommer spontant, og ofte betyder vi skal bruge mere tid på noget som muligvis allerede er lavet. Det kommer derfor enten til at koste gruppen tid og kunden penge, hvis ændringen skal foretages eller ændres på projektets omfang(scope).

Det er meget svært at beregne prisen for denne konsekvens. Da ændringerne kan gå fra at være en tekstboks design som skal ændres til store dele af systemets user interface. Vi har derfor valgt at dele 2 scenarier op og tage gennemsnittet af den pris.

Scenarie 1 – Kunden er meget kræsen omkring design og hvordan systemet håndterer data. Dette betyder kunden ønsker ¼ af programmet modificeret til hver fremvisning.
Som beskrevet tidligere har vi kontakt med kunden i meget af udviklingsprocessen, og vi viser derfor konstant dele af systemet. Vi fanger derfor de ændringer som kunden måtte have i de tidligere faser. I

dette scenarie er kunden kræsen, og ønsker konstant ændringer i de dele vi viser. Vi har i vores sprint faser estimeret os frem til at programmeringen for systemet tager cirka 70 dage.

1 dag = 8 timer
500,-(timeløn for programmør) = 4000,- pr. dag
280.000,-
280.00*0,25 = 70.000 (vi multiplicerer med 0,25 idet vi antager kunden vil have modificeret ¼ til hver fremvisning)

Scenarie 2 – Kunden er ikke kræsen omkring design og hvordan systemet håndterer data.
Her anvender vi samme måde at beregne prisen på.

1/10 (her ønsker kunden kun en 1/10 modificeret til hver fremvisning)

280.00*0,1 = 28.000

Altså bliver prisen 70.000-28.000 = 42.000,- Denne pris passer i øvrigt godt med den kunde vi antager at have med at gøre. Vores opfattelse er, at vores kunde går op i designet passer til det han har givet udtryk for, samt den performance systemet har. Vi har ikke den opfattelse af, at kunden konstant ændrer mening og vil have lavet om på noget, der allerede har en glimrende funktionalitet. Selvom vi giver udtryk for at vi kender kunden, har vi alligevel valgt denne metode at beregne os frem til, idet vi ikke er sikre på, hvordan kunden er at arbejde sammen med!

Risk Exposure: RE bliver derfor beregnet til RE = P * C
RE = 0.8*42.000
Risk exposuren for denne risiko bliver derfor = 33.600,-

Risk Identification: Projektet er mere komplekst end antaget.
Risk Probability: 30 %
Risk impact: 2
Skulle vi gå hen og opleve at projektet bliver mere komplekst end vi i starten antog, har vi valgt at estimere nogle konsulenttimer indover prisen. Konsulenten vil primært blive anvendt når vi programmerer systemet. Vi har estimeret os frem til at skulle benytte konsulent ¼ af tiden under vores programmering da vi ikke rigtigt har erfaringer med projekter som dette. Regnestykket bliver derfor:

1/4 indover. 70 dage = 8 timer. En ¼ af den tid er 140 timer * konsulent pris 1000,- altså 1000*140 = 140.000,-

Risk Exposure: RE = P * C

140.000*0,3 = 42.000,-

Regner vi de yderligere estimater ud for de resterende ”relevante” risici får vi samlet:

33.600+42.000+(140)=75.600,-

De 140.000,- kommer fra backup-risikoen (ID_03) og er udelukkende vores tab. Det er derfor ikke noget kunden skal betale!

Det skal lige nævnes, at vi selvfølgelig vil snakke med kunden om vores estimater. Det kan eksempelvis være kunden er tilfreds med, at vi ikke anvender konsulent, idet det gør at prisen for projektet stiger. Kunden kunne muligvis være tilfreds med, at vi blot ændrer scope, hvis projektet bliver for kompleks.

RMMM Planer

Herunder ses nogle af de relevante risicis informationsskemaer, som indeholder mitigation, monitoring og management. De risici vi har taget udgangspunkt i er kun risici, der er over cutoffline i vores risikotabel som blev vist i afsnit "risikotabel". De øvrige RMMM planer kan findes i bilaget. Mitigation går ud på, at man ser på hvad man skal være opmærksom på samt hvilken konsekvens risikoen kan få. Monitoring handler om hvordan vi observerer om risikoen er på vej til at indtræffe. Management handler om hvordan vi handler, hvis risikoen skulle indtræffe.

Risiko informationsskema			
Risiko ID: ID_01	Dato: 03/05-14	Sandsynlighed 80 %	Konsekvens 3
Beskrivelse Kunden ændrer krav til systemet.			
Mitigation: Hvis kunden ændrer krav til systemet, kan det få en betydning for vores deadline og dermed den pris som kunden skal betale. Det er derfor vigtigt, at vi viser kunden dele af systemet, eftersom kunden kan indskyde med de eventuelle ændringer der måtte findes. Herved kan vi ændre de eventuelle krav, som kunden måtte have, inden det vil tage for lang tid for os at modificere. **Monitoring:** Da vi afleverer dele af systemet til kunden i hvert sprint, bliver vi gjort opmærksomme på de ændringer, der eventuelt måtte være. Skulle der være findes ændringer, indgår vi en dialog med kunden og finder en løsning.			
Management/contingency plan/trigger: Skulle der findes ændringer til systemet fra kundens side, vil vi indgå dialog med kunden og finde en fælles løsning. Deadline kan flyttes, hvis dette bliver nødvendigt. Vi snakker her med kunden om en han/hun ønsker projektet som successful failure eller failure success. Med dette menes hvad kunden vægter mest i projekttrekanten. (Time,scope eller cost).			
Initiativtager: Lars Skaaning Jensen		Underskrevet: Lars Skaaning Jensen	

Risiko informationsskema			
Risiko ID: ID_02	Dato: 04/05-14	Sandsynlighed 30 %	Konsekvens 2
Beskrivelse: Projektet er mere komplekst end antaget.			
Mitigation: Da vi i gruppen ikke har arbejdet med projekter, der ligner dette, kan vi risikere ikke at gennemføre projektet. Dette resulterer i en utilfreds kunde og dermed fiasko for vores vedkommende. **Monitoring:** Vi skal holde øje med, hvordan kunden har det, med det han modtager. Vi skal yderligere have meget fokus på hvordan vores gruppemedlemmer tackler opgaven. Skulle gruppemedlemmerne mangle hjælp må vi se om vi kan hjælpe hinanden. I vores daglige møder holder vi øje med dette. Vi har også en burndown chart, hvor vi kan se om vi kan følge med i opgaverne.			
Management/contingency plan/trigger: Skulle vi falde bagud eller ikke komme videre med vores projektudvikling, må vi i samarbejde med kunden aftale, hvilken vej vi skal gå. Vi kan vælge at ændre på scope, hvilket betyder at vi kan nå det til tiden og til prisen, eller vi kan vælge at udsætte tiden og dermed øge prisen. Dette skyldes at vi må søge hjælp hos en konsulent.			
Initiativtager: Lars Skaaning Jensen		Underskrevet: Lars Skaaning Jensen	

Use Cases(Christopher)

Til dette projekt har vi valgt at bruge en tosidet use case, hvori vi placerer aktøren til venstre og systemet, som aktøren interagerer med til højre. Use casene er udstykket sådan at aktøren foretager en handling og når der sker noget væsentligt på systemets side, noteres dette som næste skridt. Der er til dette projekt 2 systemer: en webshop, hvorfra kunder kan bestille varer og et ERP-system, hvorfra salgschef og medarbejdere kan tilgå diverse funktioner tilegnet til det de bør kunne styre i forhold til kunder, samt salg og køb af varer. Mens der skal være CRUD operationer på alle funktioner i systemet vælger vi til vores use cases kun at vise udvalgte funktioner.

Måden de udvalgte use cases er udformet, gør også at vi hurtigere kan hoppe over til system sekvens diagrammer og tilstandsdiagrammer, da der i vores use cases allerede er handlinger og tilstande. Dertil vil vi også nemt kunne se hvad de ikke viste CRUD funktioner kan se ud som, hvis der laves tilstands eller system sekvens diagram over dem.

Der er til use case #1 taget udgangspunkt i en meget logisk start, med en pre-condition der blot er at åbne hjemmesiden. Use case #1 skal desuden ses som havende funktioner der er nødvendige for at andre use cases foretaget af kunde, kan foretages.

De tre aktører valgt udfylder roller der er associeret med programmerne der laves:

En kunde der vil have mulighed for at gå ind på firmaets hjemmeside og herfra kigge på og købe varer. Til foretagelse af køb har vi dog fastslået det som en nødvendighed at kunden opretter en profil på hjemmesiden, enten før eller ved købet. I vores use cases vælger vi dog kun at vise det som to separate funktioner.

Vores anden aktør er en medarbejder. Medarbejderens primære funktion er at hjælpe kunder med køb. Dertil kan det være nødvendigt for dem også at skulle indtaste en kundes oplysninger, hvorfor de kan oprette kunder. Dette sker dog ikke via hjemmesiden, men via ERP systemet, hvorfra medarbejderen har funktioner tilgængelige hvormed de kan gøre dette.

Vores tredje og sidste aktør er salgschefen. Salgschefen vil fungere som en administrator i ERP systemet og have flere funktioner tilgængelige end medarbejderen. Salgschefens funktioner inkluderer oprettelse og fjernelse af varer fra systemet, medarbejdere og kunder(CRUD).

En anden særlig vigtig funktion for salgschefen er at kunne se lagerstatus på vare og udskrive hvad lagerstatus måtte være. Dette så personen kan sikre at de har nok på lager af en vare, eller i tilfælde af at vare bliver skadet, kan opdatere antallet af leveringsklare vare på lager. Salgschefen har desuden mulighed for at gå ind og se hvilke ordre en medarbejder har hjulpet en kunde med, dette kan være både for at se om medarbejderne foretager noget og om de foretager sig ting de ikke skulle.

Use case #2.1 Opret kundeprofil
Pre-condition: Use case #1 step 1+2.

Kunde	Hjemmeside
1. Kunde vælger at oprette profil	

	2. Hjemmeside viser profiloprettelses side
3. Kunde udfylder personlige oplysninger	
4. Kunde vælger opret	
	5. Hjemmeside gør opmærksom på validering
6. Kunde åbner mail	
7. Kunde vælger at validere e-mail.	

Post-condition: profil er oprettet og valideret

Use case #2.3 Opdater kundeprofil

Pre-condition: Use case #2.1 + Use case #1 step 1+2

Kunde	Hjemmeside
1. Kunde vælger adresse oplysninger	
	2. Hjemmeside viser kundes adresseoplysninger
3. Kunde vælger at redigere oplysninger	
4. Kunde foretager ændringer	
5. Kunde gemmer ændringer	6. hjemmeside sender email bekræftelse på ændring af oplysninger

Post condition: Kundeprofil opdateret.

Use case 3.1 Opret ordre

Primær aktør: Kunde

Pre- condition: Use case #2 + Use case #1 step 1+2

Kunde	Hjemmeside
1. Kunde finder vare	
2. Kunde vælger at bestille vare	
	3. Hjemmeside beder kunde om at logge ind
4. Kunde logger ind	
5. Kunde bekræfter adresse	

	6. hjemmeside går til betaling
7. Kunde vælger betalingsform	
8. Kunde udfylder betalings oplysninger	
9. Kunde vælger at betale.	
	10. Hjemmeside beder om endelig bekræftelse af oplysninger.
11. Kunde bekræfter oplysninger.	

Post condition: kundeordre oprettet.

Use case #4.1 Opret ny vare
Aktør: Salgschef
Pre-condition : Use case #3.1 step 1

Salgschef	System
1. Salgschef vælger at oprette ny vare	
	2. System viser vare opretningsskema
3. Salgschef udfylder vare oplysninger	
4. Salgschef vælger at uploade billeder	
	5. System bekræfter upload
6. Salgschef vælger vares hovedfoto	
7. Salgschef vælger at gemme vare	
	8. System bekræfter ny vare er gemt

Post condition: Varen er oprettet.

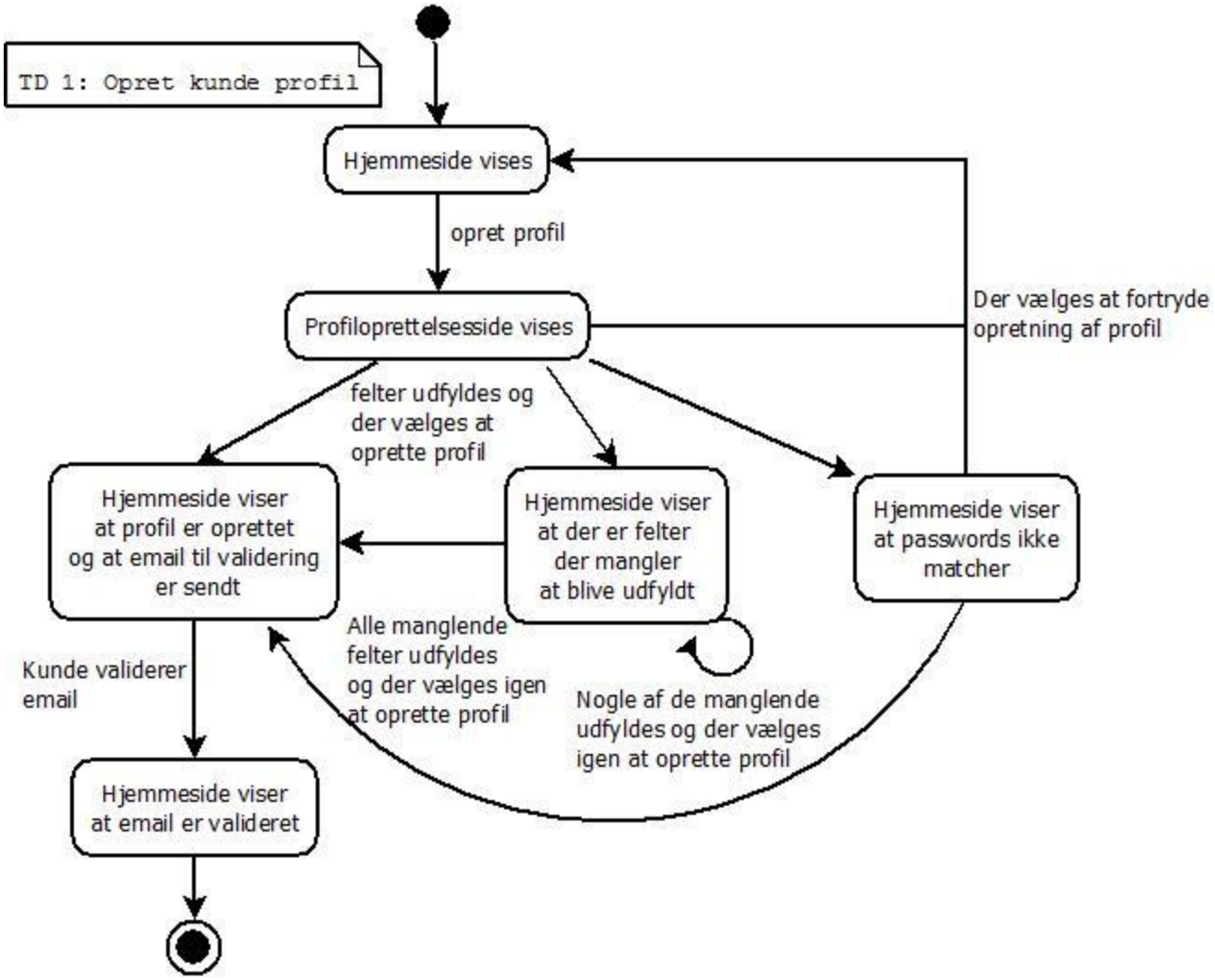
Tilstandsdiagrammer(Christopher)

Som en del af vores funktionsanalyser udarbejder vi en række tilstandsdiagrammer. Vi bruger disse tilstandsdiagrammer til at beskrive det næste skridt i udviklingen af metoderne, samt brugerfladen. I sig selv beskriver de ikke hvordan hverken metoderne eller brugerfladen kommer til at se ud, men de beskriver hvilken funktionalitet de skal besidde og hvordan de skal opføre sig. Dette inkluderer typiske fejl, som kan opstå i de processer, som aktørerne skal igennem. Vi sikrer altså at systemet ikke går ned såfremt brugerne skulle glemme at foretage en handling, skriver tekst i et felt beregnet til tal eller trykker på en knap på et forkert tidspunkt.

De indledende tilstandsdiagrammer lavet til dette projekt beskriver de højst prioriterede funktioner I systemet;

Oprettelse af kundeprofil

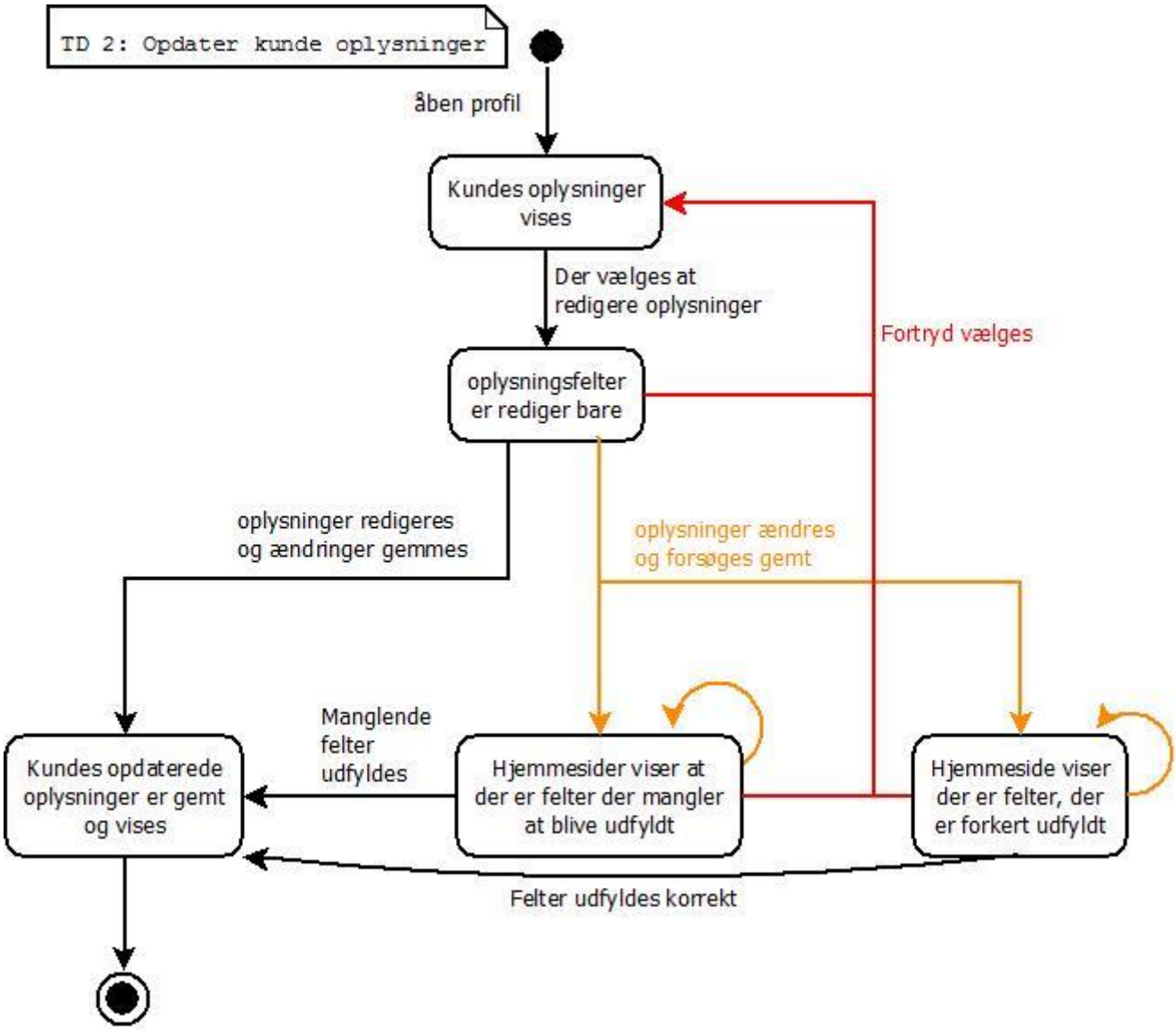
Oprettelse af kundeprofil er højt prioriteret da uden en profil, vil kunder i vores system ikke være i stand til at handle. Det er sat som et krav at for at kunne foretage et køb, skal man have en profil, hvori info omkring navn og bopæl, skal være gemt. I tilstandsdiagrammet vises som navnet antyder, hvad en kunde skal foretage sig, for at oprette en profil. Inkluderet her er de typiske fejl, som kan ske i forløbet og hvad kunden får at vide, såfremt de støder ind i fejlene. Det der er tænkt som de typiske fejl her, er at kunden enten glemmer at udfylde et felt, som skal udfyldes eller taster noget forkert i et felt. Sidstnævnte kunne typisk være bogstaver i et felt beregnet til tal, angivelse af et ukendt postnummer eller at password felternes tekst ikke er ens. En sidste funktion er retten til at fortryde, kunden skal på ethvert tidspunkt før den fulde oprettelse kunne vælge at gå tilbage, såfremt de ikke har lyst til at oprette en profil alligevel.



Opdater kundeprofil

Kunder skal være i stand til at opdatere deres profil, såfremt de har ændret adresse, telefonnummer eller andet, siden de sidst var inde at handle på hjemmesiden. For at kunder derfor ikke skal til at oprette en ny

profil, såfremt de ændrer adresse, er det vigtigt at de kan ændre deres adresse. I tilstandsdiagrammet for opdatering af kundeprofil har vi at gøre med mange af de samme informationer, som ved oprettelse. Den største forskel mellem opdatering og oprettelse er ved, at en opdatering skal du være logget ind og du skal være på din profil. Udover dette har de to funktioner de samme felter der skal udfyldes.



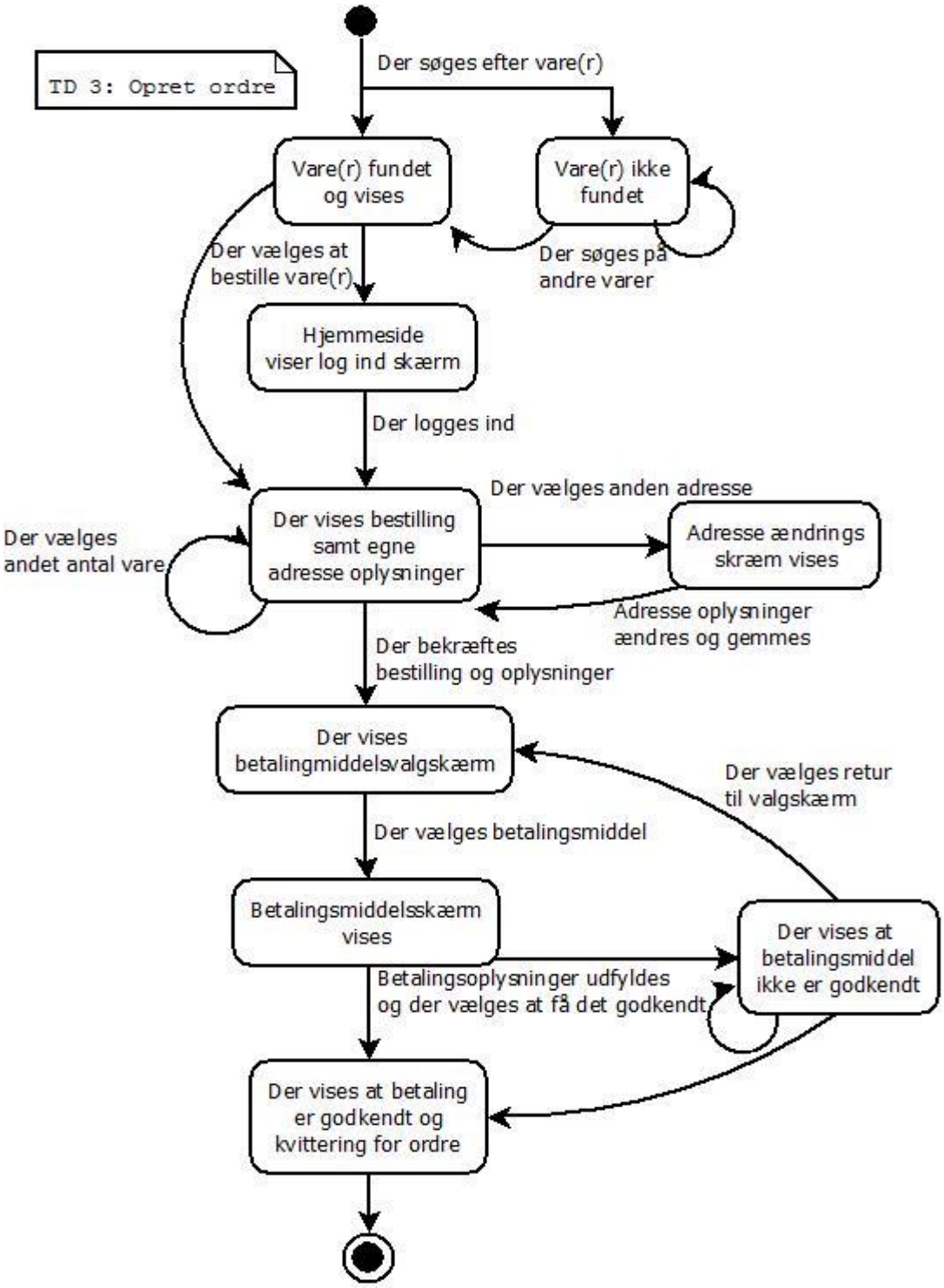
Oprettelse af ordre

Uden ordre, giver det ikke meget mening at have oprettelse af kunder eller opdatering af kundeoplysninger. Der er ingen grund til at oprette en profil, hvis man ikke har tænkt sig at købe noget af det, som firmaet sælger.

Tilstandsdiagrammet for oprettelse af en ordre er en anelse mere kompleks end oprettelse af en profil, da der ved foretagelse af et indkøb er en del flere skridt der skal tages, for at gennemføre; valg af varer og antal, login, bekræftelse eller rettelse af personlige oplysninger, valg af betalingsmiddel og endelig betaling.

Med så mange skridt er der også rigelig med mulighed for at foretage fejl og det er derfor mindst lige så vigtigt her, at kunden bliver informeret, såfremt der opstår en fejl, så de kan rette den og komme videre i deres indkøb.

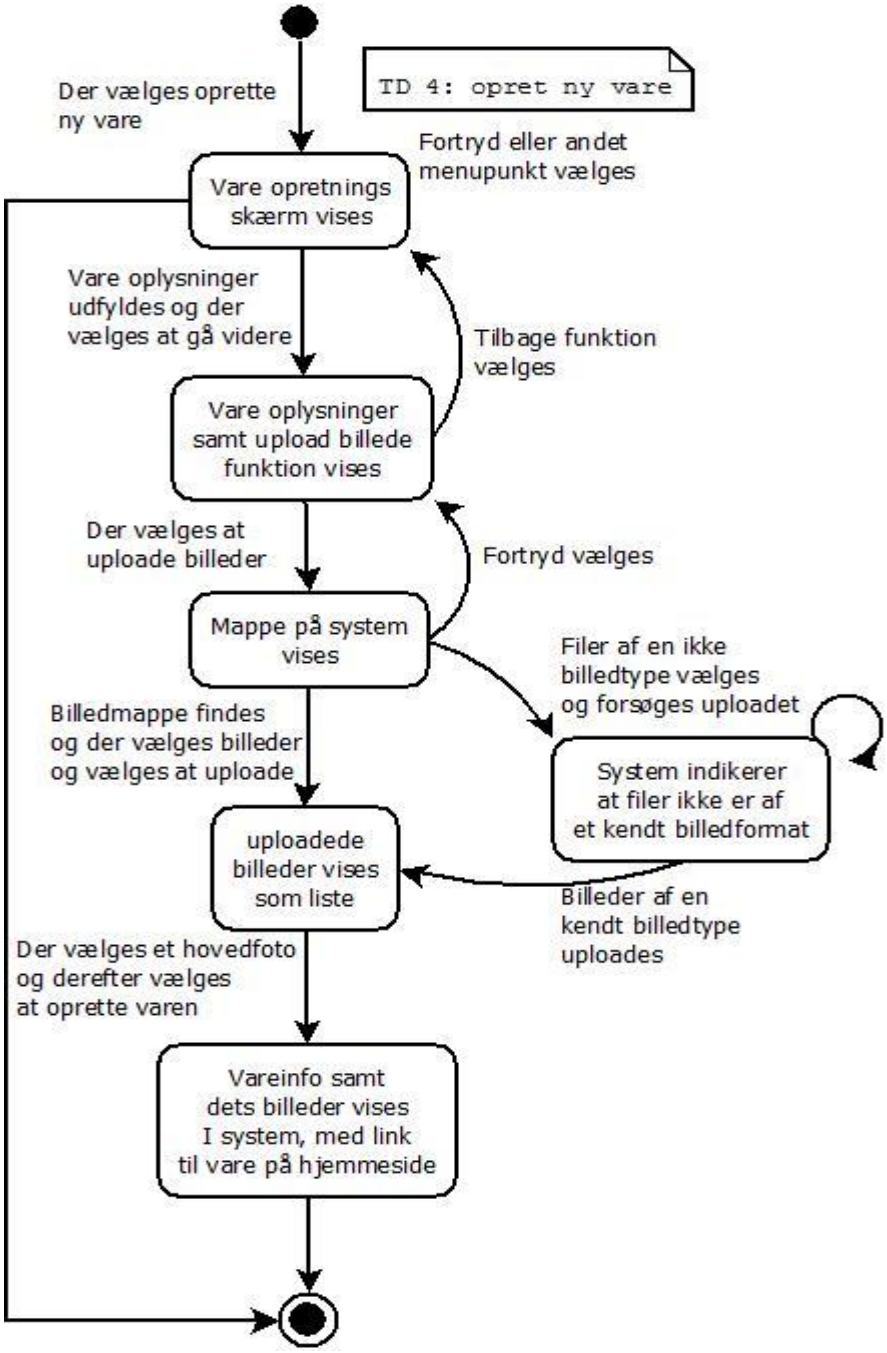
Som med de andre diagrammer har kunder også i forbindelse med oprettelse af en ordre også mulighed for at fortryde her.



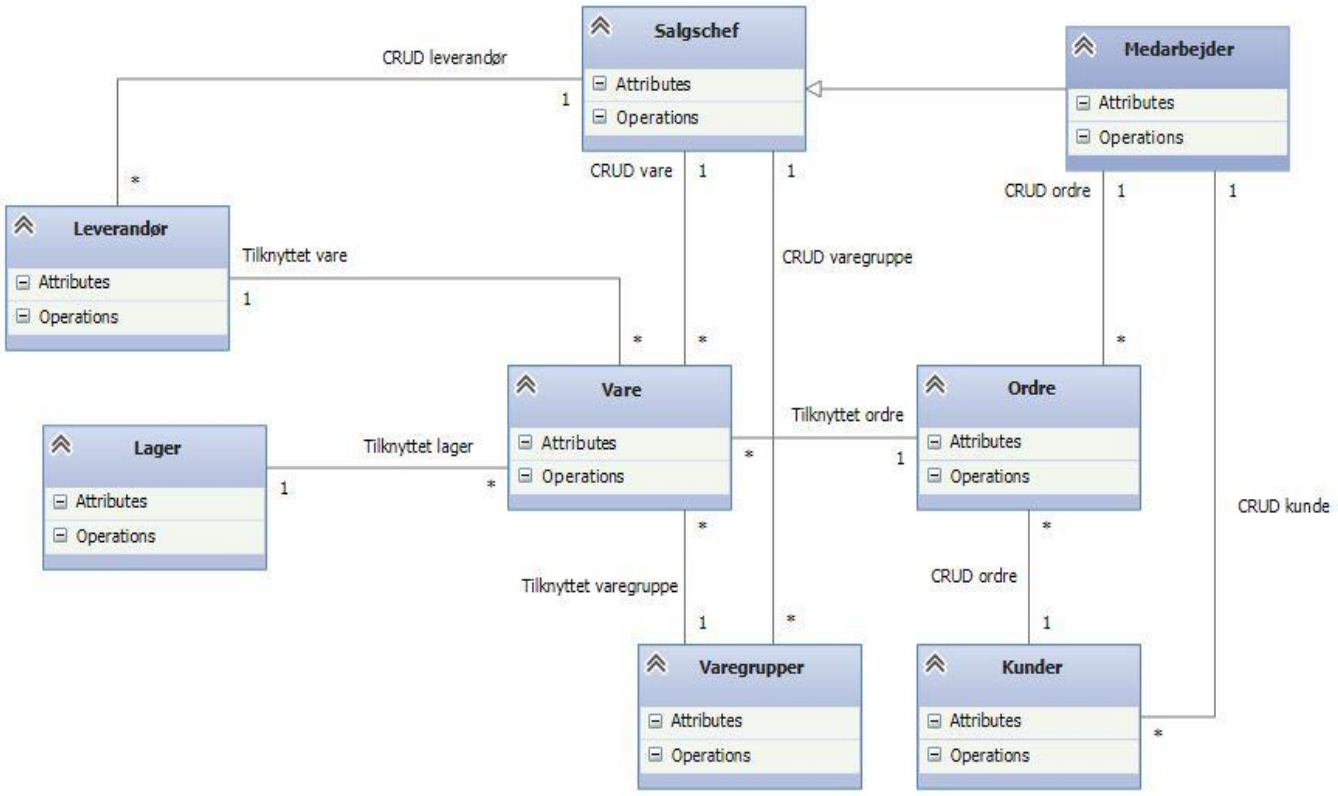
Oprettelse af vare(r)

Uden nogen varer i systemet, er det ikke muligt at foretage et indkøb, hvilket så gør oprettelse og opdatering af profil redundant.

Til oprettelse af en vare, skal vi have fat i en anden person og benytte et ERP system, som tilknyttes hjemmesiden. Funktionaliteten bagved er dog meget ala det der er på hjemmesiden, den største ændring kommer i muligheden for at uploade billeder, som systemet skal kunne genkende og fremvise. Desuden skal systemet kunne indikere hvis det salgschefen har prøvet at uploade ikke er af et (af systemet) kendt billedformat. Mulighederne for at gå tilbage, fortryde oprettelsen og generelle fejlmeddelelser, såfremt noget udfyldes forkert eller mangelfuldt, går igen fra de tidligere diagrammer.



Domænemodel(Lasse)



Vi har lavet en domænemodel over vores ERP - salgssystem for, at danne os et bedre overblik over hvilket klasser vi skal bruge i vores system og for at give os et bedre overblik over systemet.

Vi har lavet en specialisering af vores medarbejder, da det kun er salgschefen der kan oprette en vare og leverandør.
Det er muligt for en kunde og medarbejder at oprette en ordre som har nogle varer som tilhører i en varegruppe og er tilknyttet et lager.

Systemsekvensdiagrammer(Christopher)

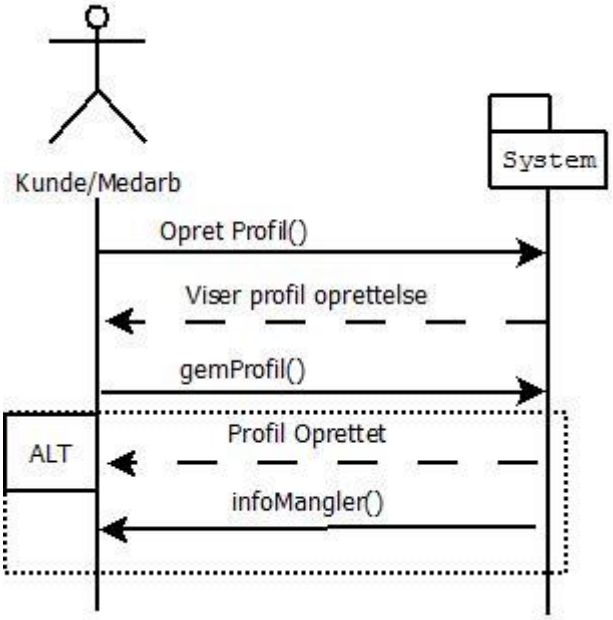
Efter vores tilstandsdiagrammer, som gav et godt indblik i processerne, som aktørerne har mulighed for at foretage sig, i forbindelse med udførelse af diverse opgaver, går vi videre til system sekvens diagrammer(SSD). I disse, går vi skridtet videre og viser de valgte metode navne og hvornår de interagerer med systemet. Det vil desuden tydelig fremgå, hvornår funktioner vil være indkapslet i loops, alternates eller er muligheder, som aktøren blot kan vælge at gøre.

De udvalgte SSD'er, er baseret på vores tilstandsdiagrammer, da vi igen finder at dette er de højest prioriterede funktioner, som skal laves i de tidlige sprint.

Opret kundeprofil

Aktøren til oprettelse af kundeprofil er kunder og medarbejder. Medarbejdere er medtaget, da de via ERP-systemet skal være i stand til at oprette profiler for kunder, som de har haft kontakt med enten via telefon eller i butikken.

I SSD'en har vi de tre metoder, som indgår i oprettelsen af en profil, den første metode opretProfil() fører aktøren til profil oprettelses skemaet, hvilket herefter skal udfyldes. Når dette er gjort vil aktøren gemme informationen, dette sker ved et kald til gemProfil(). Metoden gemProfil() kan så enten svare tilbage at profilen er oprettet eller melde fejl, hvilket vil føre til infoMangler() metoden. Denne vil så tage stilling til hvorledes der er fejl i profiloprettelsen og melde til aktøren, hvad der måtte mangle eller være udfyldt forkert.

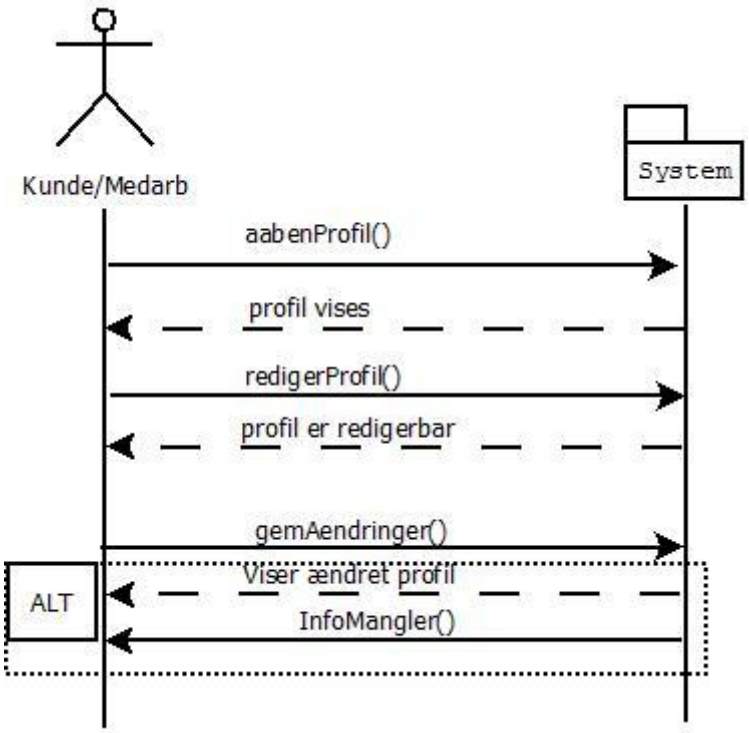


Opdater kundeprofil

Opdatering af profil har den pre-condition at aktøren er logget ind, uden dette kan det ikke lade sig gøre at komme ind på deres profil.

Som ved vores tilstandsdiagram, minder denne SSD også meget om scenariet til opret profil, blot med den væsentlige forskel at du har her en profil og har valgt at logge ind og redigere oplysningerne i den, frem for at skulle udfylde dem fra ny.

Metoden redigerProfil() vil kalde informationen frem for den aktør der er logget ind eller valgt og gøre dette redigerbar. Herefter kan aktøren vælge at ændre oplysningerne og gemme dem ved hjælp af gemAendringer() metoden. Igen har vi en mulighed for at der kan være information, som er udfyldt forkert eller mangler og disse tilfælde skal metoden infoMangler(), som er beskrevet under oprettelse af kunde profil kaldes. Metoden genbruges dermed i dette scenarie.

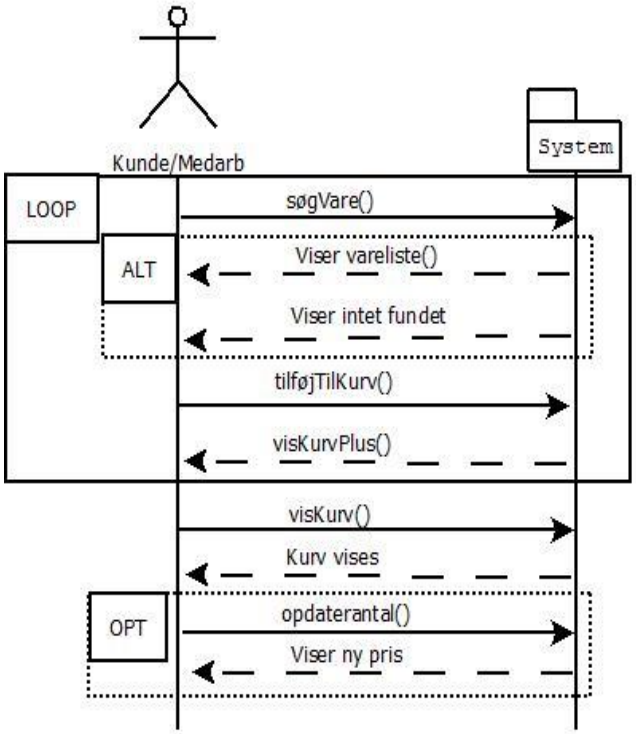


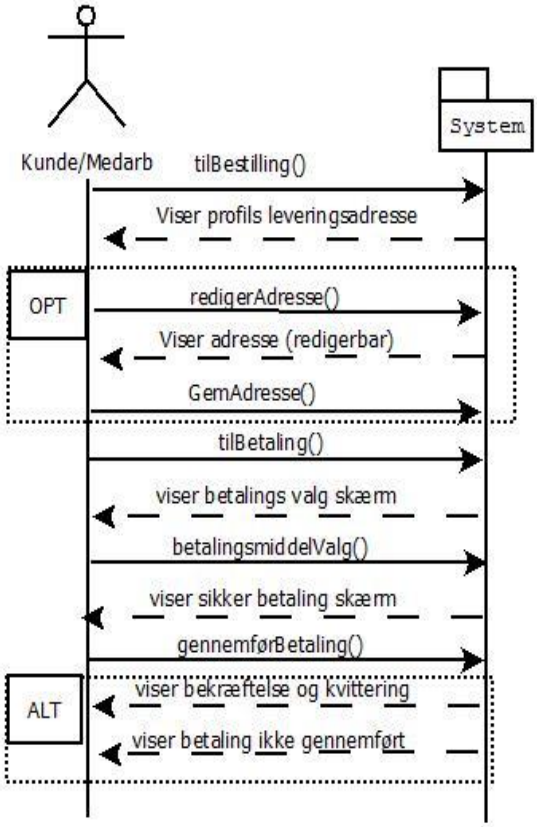
Oprettelse af ordre

Oprettelsen af ordre har vi tidligere beskrevet som en lang proces. I SSD'erne vælger vi derfor at dele denne op i to diagrammer, med den ene som pre-condition. Dette gøres for at mindske kompleksiteten af diagrammet og for at vise at den første kan gøres uafhængigt af den anden, såfremt der i senere tilfælde skal aktører ind, som vil bestille på anden vis end private kunder.

Diagrammerne, som ordre oprettelsen er delt op i, er valget af vare og bestillings delen.

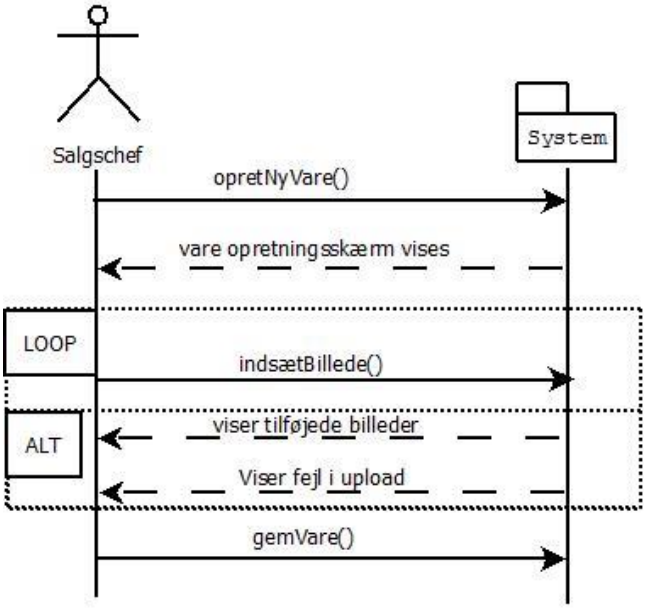
I det første diagram, hvor aktøren foretager søgning og valg af vare, er dette omkredset af et loop. Dette gøres fordi aktøren ikke skal tvinges til at gå videre til bestillingen før de har valgt alle de varer, som de måtte ønske at bestille. Denne del indeholder også et alternativt scenarie, da det ikke er garanteret at der ved en søgning kommer et resultat tilbage. Hvorvidt dette er fordi aktøren har stavet varens navn forkert eller blot at varen ikke eksisterer, tages der ikke hensyn til. Vi har i dette system ikke tænkt os at implementere rette mekanismer, der vil spørge ind til, hvorvidt det var noget lignende, som aktøren mente.





Oprettelse af vare

Oprettelsen af en vare er ikke en kompliceret proces og den minder igen meget om hvad der foregik på tilstandsdiagrammet. Til forskel er der dog her ikke en hændelser taget med, som ikke har med funktioner at gøre. Som i de foregående SSD'er betyder dette at der ikke medtages hændelser, som ikke involverer aktivering af en metode.

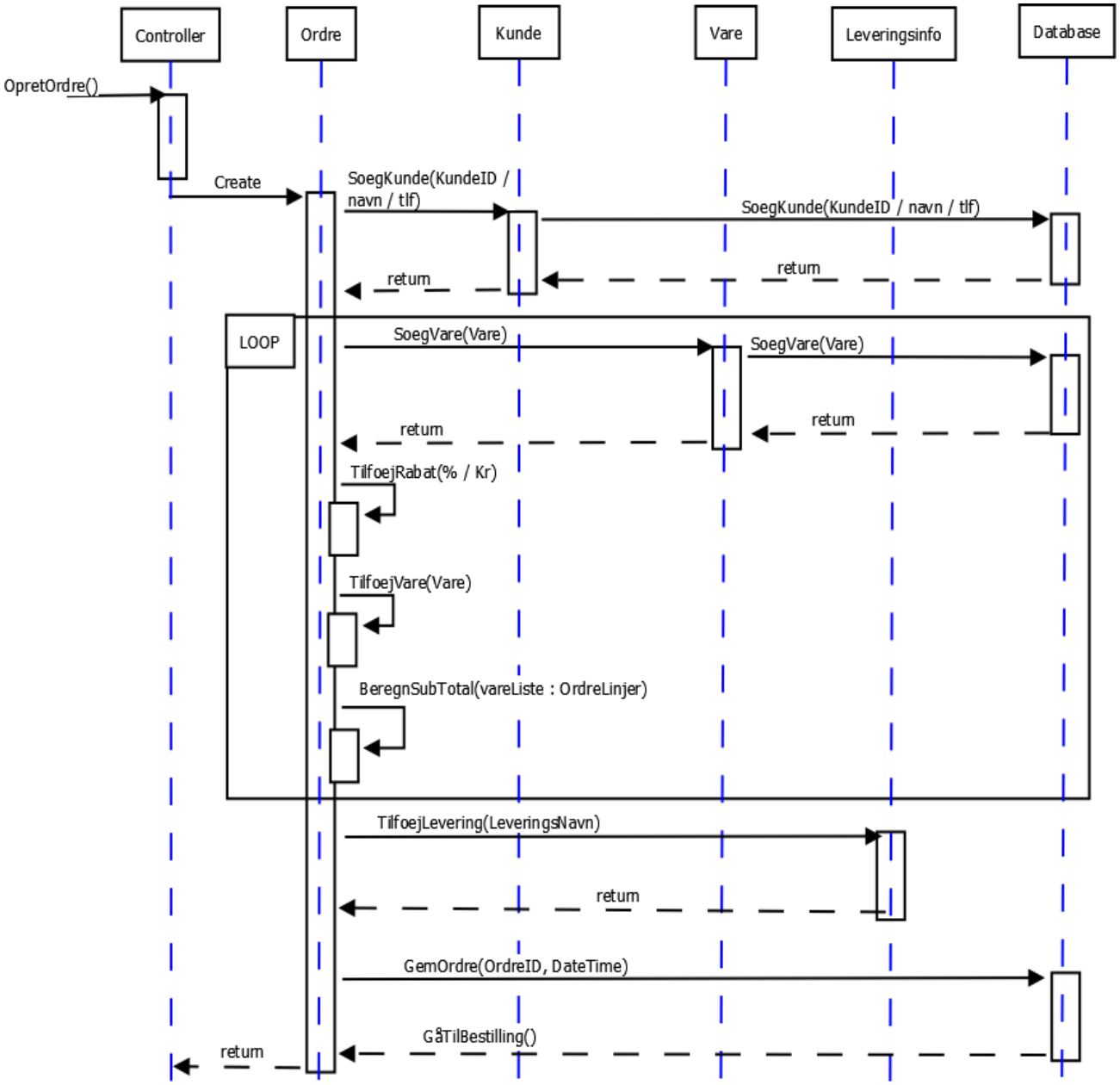


Sekvensdiagram (Tobias)

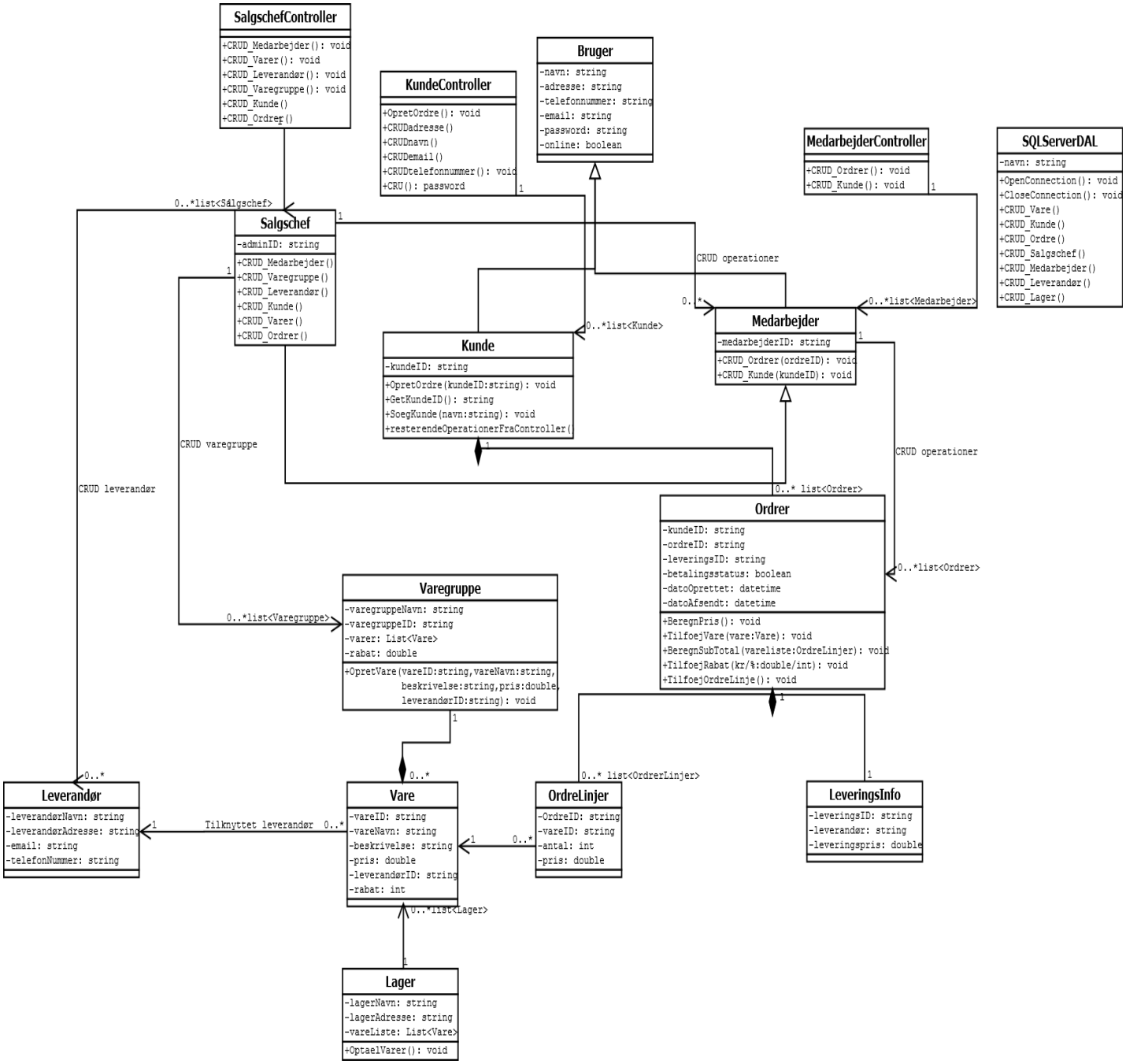
Til yderligere at supplere arkitekturen er der blevet lavet sekvensdiagram over det mest væsentlige i første inkrement, oprettelse af en ordre. Da vi både skal tage stilling til hvordan dette foregår på websiden, hvor det er kunden der opretter ordren, og i salgsmodulet, hvor det er en medarbejder, er der blevet lavet 2 versioner.

Pre-condition: Kunden er oprettet i systemet før ordren oprettes.
Første version fokuserer på salgsmodulet hvor aktøren er en medarbejder. Først tilføjes en kunde til ordren: Kunden tilføjes til ordren, da langt de fleste ting der sælges af virksomheden er dyre mærkevarer, dog med undtagelse af brugskunst og deslige. Derefter tilføjes varer til ordren, og hver gang der tilføjes en ny vare, beregnes der automatisk en ny subtotal på ordren. På de enkelte varer eller hele ordren kan der gives x kroner eller procenter i rabat. Når medarbejderen så er færdig med at tilføje varer, bliver der tilføjet en leveringsmetode, som vil være prædefineret. Dernæst bekræftes ordren, der autogenereres et ordre ID og en dato for ordren, og man vil blive viderestillet til et betalingsvindue. Kunderne kan få en faktura med hjem og betale via netbank, og først når betalingsstatus er ”Betalt”, er ordren godkendt og varer(ne) fjernes fra lageret.

Anden version fokuserer på websiden, hvor aktøren er en kunde. En kunde på en webside vælger oftest et af to scenarier: kunden logger ind på websiden eller kunden tilføjer varer til kurven. I vores sekvensdiagram har vi taget udgangspunkt i, at kunden tilføjer de varer der ønskes, og så fortsætter til bestilling. Her skal kunden logge ind hvis dette ikke allerede er sket, og dernæst vælge nuværende adresse som leveringsadresse eller indtaste en ny. Derefter skal kunden vælge betalingsmiddel, hvorefter kunden kan bekræfte ordren og gå videre til betaling. Først når betalingen er gået igennem er ordren godkendt og ordren oprettes med ordre ID og en dato.


Controller
Ordre
Kunde
Vare
Leveringsinfo
Database
OpretOrdre()
Create
SoegKunde(KundeID / navn / tlf)
SoegKunde(KundeID / navn / tlf)
return
return
LOOP
SoegVare(Vare)
SoegVare(Vare)
return
return
TilfoejRabat(% / Kr)
TilfoejVare(Vare)
BeregnSubTotal(vareListe : OrdreLinjer)
TilfoejLevering(LeveringsNavn)
return
GemOrdre(OrdreID, DateTime)
GåTilBestilling()
return

Klassediagram(Lars)



For at forklare de overvejelser, der er blevet lavet i diagrammet, har vi anvendt GRASP. Bemærk i øvrigt at dette klassediagram er blevet udarbejdet i inkrement 1, hvilket betyder at det ikke er endeligt!

Controller

Vi har valgt at lave tre controllere for at uddelegere de opgaver, som aktørerne kan foretage sig. Dette er i øvrigt også med til at sikre at vi ikke risikerer at tilgå metoder, som aktøren ikke har tilladelse til. Et eksempel på dette kunne være en kunde, der opretter en salgschef. Står der ikke i KundeControlleren at en

kunde må oprette en salgschef, kan kunden ikke komme til det. Bemærk i øvrigt vores anvendelse af forkortelserne CRUD. C = create, R = Read, U=Update og D står for delete. At skrive alle de forskellige metode navne på klasserne er noget vi vil have gjort, når vi er engang ville blive endeligt færdige med projektet.

Creator

Vi har i vores system mange creators. Nogle creators, der findes i vores system, er klassen Kunde, idet en kunde creator en ordre. Ligeledes er klassen ordre også en creator, idet den creator både leveringsinfo og ordrelinjer. Salgschef er creator for medarbejder, kunde, ordrer, varegruppe, vare og leverandør. Salgschefcontrolleren er i øvrigt creator af salgschef. Det gælder for alle controllers, at de creator deres objekter, som de hører sammen med, hvilket ses i diagrammet.

Information Expert

Vi har i vores klassediagram mange information experts. Der kan nævnes klassen ”Ordre”. Denne klasse indeholder data fra ordrelinjerne, og kan derfor beregne en total pris af dette. Det bør være klassen ”Ordre”, der indeholder denne metode, fordi den er ansvarlig for at beregne dens pris. Et objekt skal kunne tage vare for sig selv, dermed mindsker vi kobling og øger kohæsion. Klassen ”lager” er et andet eksempel på en information expert. Denne klasse ved hvor mange varer, der er på lageret. Altså er det oplagt at det er lager, som tager ansvaret for at optælle de varer, der måtte være på lageret.

Kohæsion

Vi har under udarbejdelse af klassediagrammet taget højde for kohæsionen. Med dette menes der at vi har holdt metode navne til de klasser, hvor det giver mest mening at have dem.

Kobling

Koblingen i vores system er så minimal, som vi kunne gøre det på nuværende tidspunkt. Dette betyder ikke, at der ikke kan findes en kobling i vores system, men den kobling, der findes er nødvendig! Vi har muligvis en for høj kobling mellem vores ”SalgschefController” og ”Salgschef” klassen. Grunden til dette er, at når vi ændrer et navn på en metode eller opretter en ny, skal ændre dette i salgschefklassen. Vi har lavet controllerne for at arbejde med ET objekt. Dette objekt(eksempelvis en medarbejder) modtager data fra UI-laget og sender herefter dataen videre ned i domain-laget, hvor dataen vil blive behandlet. Vi ville muligvis ændre noget på denne kobling i en senere fase i vores projektudvikling.

EER-model (Tobias)

Der var enighed blandt teamet til at skitsere en EER-model for at supplere klassediagrammet. Denne EER-model dækker over salgsmodulet i ERP-systemet og skal være med til at synliggøre processen bag et salg af en vare.

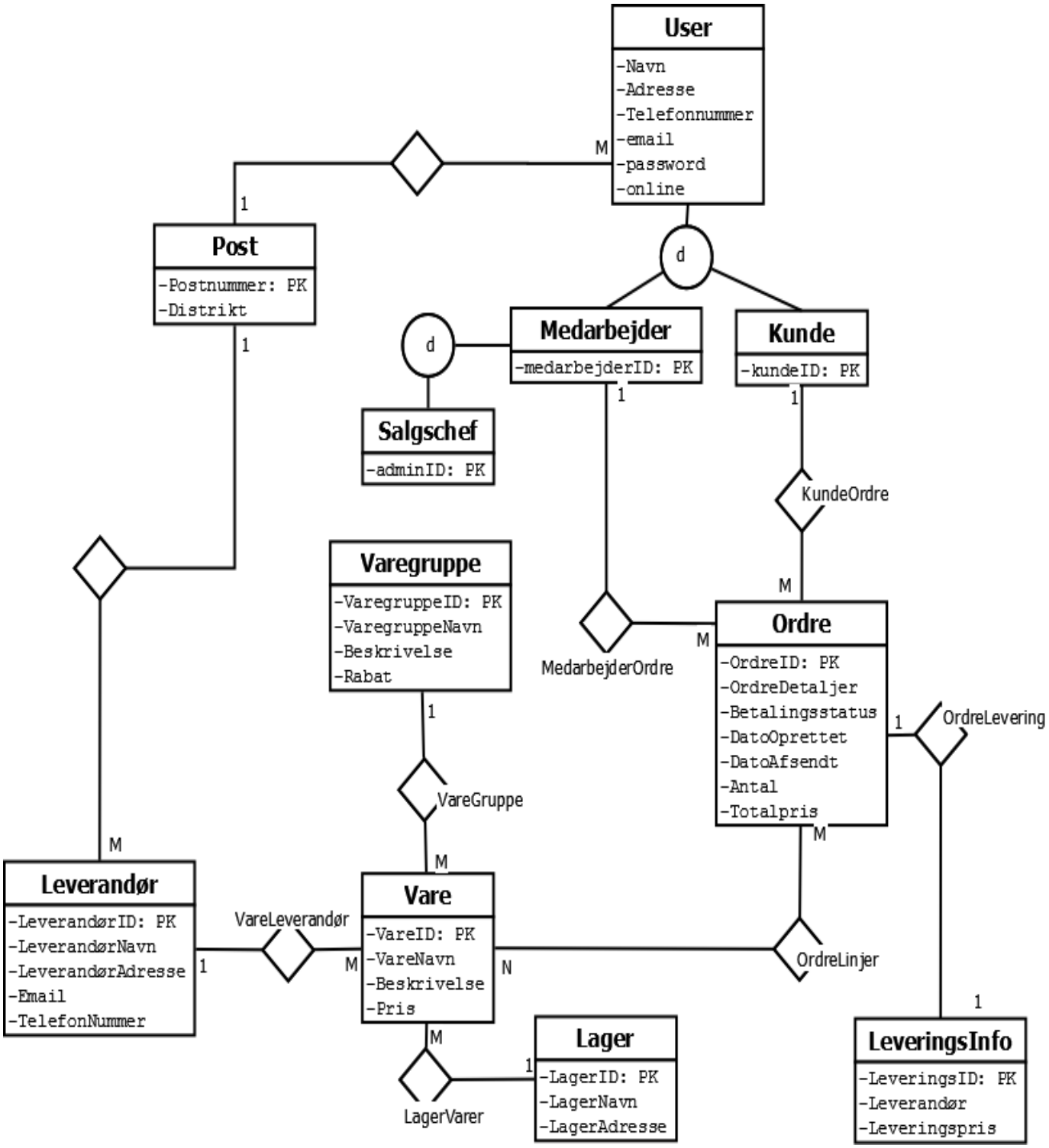
EER-modellen tager udgangspunkt i en bruger, som enten er specialiseret som kunde eller medarbejder. Medarbejder kan så yderligere være specialiseret som salgschef, men i denne prototype af systemet findes der kun én salgschef. Det vil typisk være salgschefen der opretter nye varer, varegrupper og leverandører, men da vi arbejder med én salgschef, er det ikke nødvendigt at have relationer til disse tabeller.

Medarbejder er tilknyttet ordre, da kunden ønsker gennemsigtighed i systemet, så det er muligt at se hvilken medarbejder, der har oprettet hvilken ordre for en given kunde. Det sker når både et medarbejder ID og et kunde ID havner på ordren. Der er dog ikke nødvendigvis knyttet en medarbejder på en ordre, da kunder selv kan oprette ordrer via webshoppen og gennemføre køb.

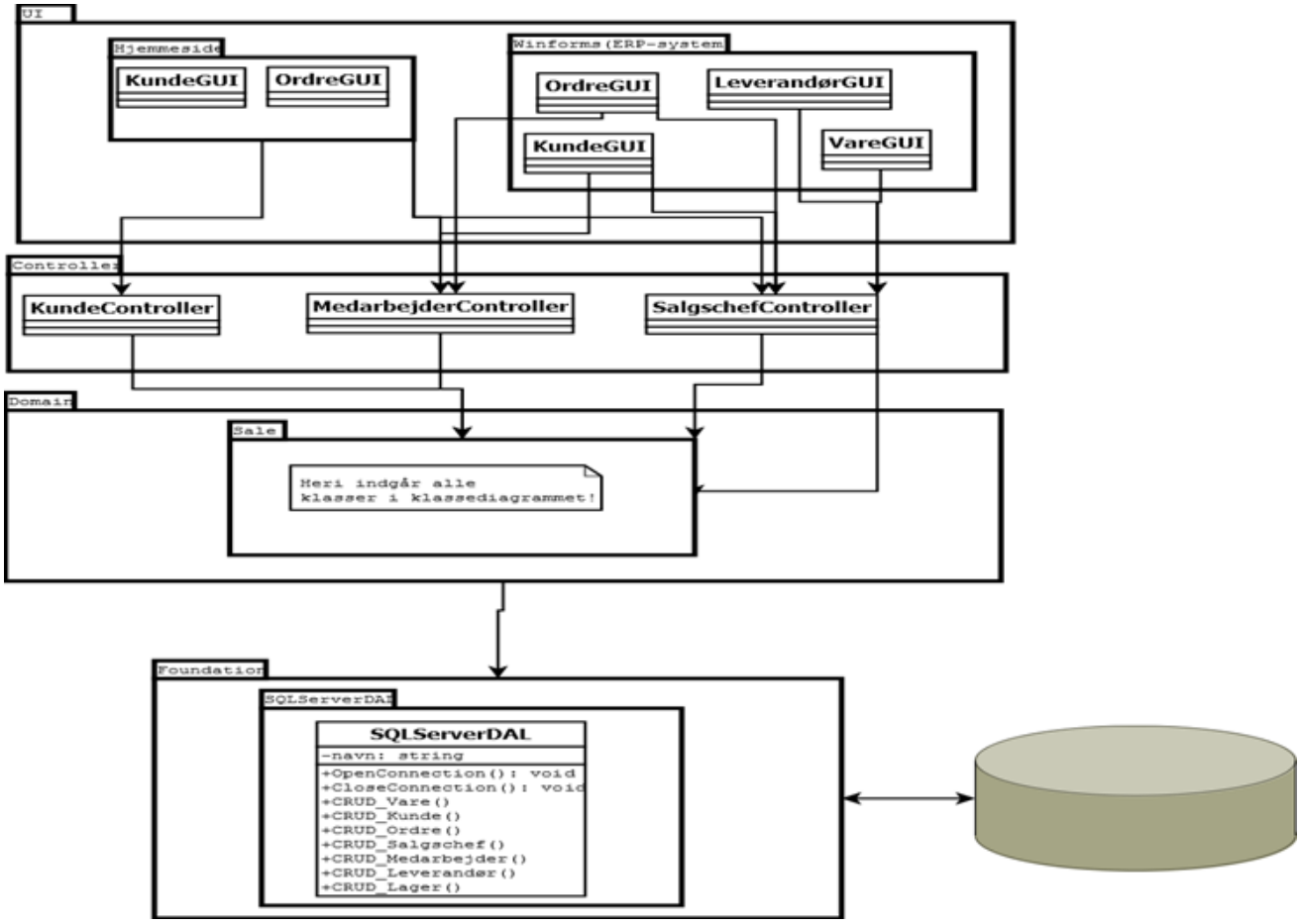
En ordre gives yderligere en tilknytning til ordrelevering, da firmaet arbejder med faste leverandør firmaer (PostDanmark og GLS). Ordren er også knyttet til varetabellen med relationen ordrelinjer, hvor man kan se hvilke varer der bliver solgt i en ordre. Ordrelinjer vil under mapning af EER-modellen blive til en tabel for sig selv.

Noget af det mere væsentlige for kunden er varer- og lagerstyring. Vi har valgt en klassisk løsning med et lager, hvor der er mulighed for at tilføje flere lageradresser hvis kunden har brug for mere lagerplads. Dette gør det også nemmere at overskue hvor en given vare befinder sig. Desuden er varer tilknyttet én varegruppe, som der kan gives rabat på, mens der også kan gives rabat på enkelte varer.

Firmaet har aftaler med leverandører af varer, og derfor har en vare en fast leverandør. Derfor har firmaet også bedt om mulighed for at katalogisere leverandører med relevante attributter. Leverandører og brugere er tilknyttet en posttabel med distrikter for at gøre nemmere ved oprettelse af brugere og leverandører.


User
-Navn
-Adresse
-Telefonnummer
-email
-password
-online
M
1
Post
-Postnummer: PK
-Distrikt
d
Medarbejder
-medarbejderID: PK
Kunde
-kundeID: PK
Salgschef
-adminID: PK
KundeOrdre
Varegruppe
-VaregruppeID: PK
-VaregruppeNavn
-Beskrivelse
-Rabat
MedarbejderOrdre
Ordre
-OrdreID: PK
-OrdreDetaljer
-Betalingsstatus
-DatoOprettet
-DatoAfsendt
-Antal
-Totalpris
OrdreLevering
VareGruppe
Leverandør
-LeverandørID: PK
-LeverandørNavn
-LeverandørAdresse
-Email
-TelefonNummer
VareLeverandør
Vare
-VareID: PK
-VareNavn
-Beskrivelse
-Pris
OrdreLinjer
Lager
-LagerID: PK
-LagerNavn
-LagerAdresse
LagerVarer
LeveringsInfo
-LeveringsID: PK
-Leverandør
-Leveringspris

Arkitektur (Lars)



Her ses et billede af, hvordan vores arkitektur for programmet er:

Som det fremgår af arkitekturen anvender vi MVC(model, view, controller). Vi har dog tilføjet et nyt lag kaldt ”foundation”. Dette lag kommer med inspiration fra PCMEF(Presenter, Controller, Mediator, Entity, Foundation) arkitekturen.

Design og usability(Lasse)

Brugervenlighed /Usability (ERP - salgssystem)

Skærmbillederne findes i bilag.

Vores betaversion på brugergrænsefladen til det nye system er udarbejdet med stort henblik på, at de samme knapper er placeret samme sted på de forskellige skærmbilleder, og at alle knapper og felter ved hjælp af teksten hertil fortæller præcis hvad de gør.

For eksempel sidder knapperne ”Forside” samme sted på alle skærmbillederne, hvor det er muligt at komme tilbage til forsiden. På alle de sider, der ikke er ”forsiden”, er der en ”Luk”-knap placeret i bunden af skærmbilledet, nederst til venstre på pop-ups. Dette gør, at brugeren altid kan afbryde, og komme tilbage til foregående skærmbillede.

Knapper og felter der har noget med hinanden at gøre, er også placeret forholdsvis tæt sammen, så de opfylder gestaltloven om nærhed. På ”Kartotek Varer”-billedet er søgeresultatet for det søgte varenummer f.eks. placeret tæt og lige overfor hinanden, så de opfattes som et element. Kartotek Varer er også omkranset med en ramme som opfylder gestaltloven om lukkethed og ordrelisten også opfattes som et selvstændigt element.

Brugergrænsefladen er også udarbejdet ud fra kriteriet om at det skal være nemt at bruge og lære. Det er overskueligt, at se hvad hvert felt og knap gør, og det er godt med plads omkring de enkelte elementer, så det ikke er forvirrende eller sammenpresset at kigge på.

Systemets brugergrænseflade opfylder Jacob Nielsen 10 heuristikker mere eller mindre:

1. Systemet fortæller vha.pop-ups om der er fejl eller mangler i f.eks. noget indtastet data og når en kunde, vare el.lign. er gemt, så der ikke bare trykkes på en gem - knap, uden brugeren får at vide om det indtastede er gemt eller mangelfuldt.
2. Der er ikke nogen fremmedord, sætninger eller begreber i vores system og det er meget overskueligt, hvordan systems rækkefølge for de forskellige processer er.
3. Alle vores skærmbilleder undtagen forsiden har enten en luk - knap eller en forside - knap, så det er altid mulig for Kunden at komme tilbage til skærmbilledet inden.
4. Knapper og links, der har samme funktion ser ens ud. Alle knapper er lige høje og tekststørrelsen på labels er ens. Der er ingen forvirrende farver eller forstyrrende grafik.
5. For at undgå at vare, kunder, osv. kan blive slette i vores system ved en fejl, kommer der altid et popup vindue med om man er sikker på, at man vil slette denne vare og vare kan kun blive slettet hvis kunden trykker "Ja".
6. Hvis der skal ændres i en kunde, vare, osv. søger man først på varen og derefter dobbeltklikker på varen, hvorefter man trykker på opdatere og alle oplysninger dukker op i et nyt popup vindue, så brugen er fri for at indtaste alle oplysningerne igen.
7. Da vores system ikke er særligt stort, bliver heuristik 7 ikke opfyldt, da der ikke er muligt at lave personlige genveje og lignende i vores system. Da vores system også er enkeltbrugersystem, vil det heller ikke være alle medarbejder der finde alle genveje lige vigtige.
8. Systemet er minimalistisk sat op og der er kun de funktioner, der er brug for.
9. Systemet melder hvis der mangler at blive indtastet oplysninger i et felt eller hvis der er fejl i indtastningen og derefter fortæller medarbejderen hvad der mangler.
10. Det er muligt at søge vare, ordre og kunder i vores system ud fra forskellige oplysninger. Så det er altid muligt at finde en ordre, kunde, vare ud fra få oplysninger.

For at teste vores brugervenlighed i vores system kan vi benytte kort test, brugertest og eksperttest.

Kort Testen benyttes tidligt i forløbet og kan allerede laves i forbindelse med at skitserne til brugergrænsefladen bliver lavet. En testperson får uddelt nogle papkort med de væsentligste overskrifter for systemet og skal herefter sortere nogle andre kort ind under de givne overskrifter. Overskrifterne ville i dette system være: ”Ordre” og ”Kunde”. På de andre kort kunne der f.eks. stå diverse felter fra ”Kundeoplysninger”, ordrenummer på en ordre, en vares navn, osv.

Ud fra kort testen kan man evaluere om strukturen i systemet er overskuelig, logisk og sammenhængende.

Når systemet har alle de funktioner, som vores salgsmodul har ønsket er det muligt, at lave en brugertest over om betaversionen virker og hvilket fejl og mangler der er.

Efter en kort præsentation af systemet for medarbejder, kan man opstille en række opgaver ud fra de opstillede use cases for dem og ud fra disse opgaver kan man opstille en række SMART - Mål som kunne være.

Målene er lavet for medarbejder:

- 5 ud af 5 skal kunne oprette en bruger på 3 minutter.
- 5 ud af 5 skal kunne oprette en ordre i vores system.
- 5 ud af 5 skal kunne søge en kunde og opdatere kundens oplysninger på 2 minutter.

Man kan selvfølgelig opstille flere SMART - Mål ud fra opgaverne.
I forbindelse med brugertest kan der også udføres eksperttest, der også vurderer systemets kvalitet.
En ekspert test foretages ud fra Jacob Nielsens 10 heuristikker.

Brugervenlighed/ Usability (hjemmesiden)

Skærmbillederne findes i bilag.

Ligesom vores salgssystem har vi også lavet vores hjemmeside med henblik på, at alle knapper er placeret samme sted på de forskellige sider.

Alle vores sider er lavet ud fra en skabelon, så det er lettere for kunden at finde rundt på hjemmesiden.

Vi har også lavet at hvis man trykker på virksomheden logo, komme man tilbage til forsiden. Logoet findes derfor på alle siderne på vores hjemmeside.

Vores indkøbskurv er også placeret i toppen af vores hjemmeside i højre hjørne. Dette gør at kunden altid kan se hvor mange varer der er i indkøbskurven.

Så her har vi opfyldt loven om lighed og det medvirker til, at gøre det letter for kunden at navigere på hjemmesiden.

Vi opfylder også gestaltloven om nærhed, da vores links og knapper er tæt placeret.

F.eks. står vores links i vores header op af hinanden og det er med til at gøre det lettere for kunden at overskue siden.

Hvis kunden vælger, at trykke på "Møbler" i vores header links, vil der automatisk komme links og valgkriterier fremover i vores menu links.

Loven om lukkethed bliver også opfyldt, da vores menu links er i venstre side og er aflukket fra resten af siden. Dette giver igen kunden en bedre overblik over, hvilket mærke, mindste og max pris skal være.

Hvis vi kigger på Jacob Nielsen 10 heuristikker vil punkterne for vores hjemmeside være.

1. Systemet fortæller hvis en bruger mangler indtastning i felter for CRUD bruger eller hvis brugen skal betale for en ordre. Her vil de tekstbokse der ikke er udfyldt blive markeret med rød skrift og fortælle at tekst mangler. Systemet vil også fortælle når en bruger er oprettet eller en

betaling er gennemført. Hvis bruger vil slette sin profil eller ordre vil systemet også spørger brugeren, om brugeren er helt sikker på om ordre eller sin profil skal slettes.

2. Ligesom i vores ERP - salgssystem er der ikke nogen fremmedord på forsiden, men det kan forekomme i beskrivelsen af vores varer. Processen for køb er ligesom alle andre hjemmesider. Man lægge varer i indkøbskurven og går til kassen når man er klar til at betale.
3. Da det er den samme skabelon vi bruger på vores hjemmeside, vil brugeren altid kunne komme rundt på hjemmesiden uden, at skulle klikke ind på flere sider for at komme til den side brugeren ønsker. Der vil også være mulighed for at gå tilbage gennem betalingsprocessen.
4. 4. Præcis det samme som i vores ERP - system.
5. 5. Ligesom jeg skrev i punkt 1 vil denne besked komme frem når en bruger vil slette sin profil eller ordre.
6. Kundens profil vil automatisk blive videreført til leveringsinformationer ved køb af varer og kundes varer vil også følge med uden, at kunden skal tilføje dem igen.
7. Vores system opfylder ikke heuristik 7, da brugeren ikke kan lave nogle personlige genveje eller tilpasse siden til hans ønsker.
8. Hjemmesiden har kun de funktioner som er nødvendig og har ikke nogen ekstra funktioner.
9. Ligesom i vores ERP - system fortæller hjemmesiden hvis en fejl opstår som kan ske ved indtastning af profiloplysninger eller ved betaling.
10. Der vil står i de forskellige felter hvad brugeren skal indtaste i de forskellige tekstbokse. Der er også en søgefunktion, hvis brugeren ønsker at finde en bestemt vare.

Test vil blive det samme som til vores ERP - salgssystem den eneste forskel vil være vores SMART - Mål.
De vil nok i stedet være:

- 5 ud af 5 skal kunne oprette en profil på 3 minutter.
- 5 ud af 5 skal kunne finde en bestemt vare på 1 minut.
- 5 ud af 5 skal kunne lave en søgning på en være ved brug af valgkriterier på 2 minutter.
- 5 ud af 5 skal kunne tilføje 2 bestemte vare i indkøbskurven og gå til kassen og betale en ordre på 7 minutter.

Scrum sprint(Lasse)

Resten af sprint findes i bilag.

Sprint 1 **14 dage.**
UC: #2.1 Opret kundeprofil
Produkt backlog:
- Skærmbilleder til forsiden, opret kundeprofil, kundeoplysninger på hjemmesiden.
- Skærmbilleder til forsiden, opret kundeprofil, kundeoplysninger i vores salgssystem.
- Der skal også laves et log in for kunden.
- Der skal oprettes en database med en kundetabel, hvor kunde profilen bliver gemt når den bliver oprettet og vores log in henter oplysninger fra. 8
Sprint 2 **14 dage**
UC: # 3.1 Opret ordre (kunde)
Produkt backlog:
- Skærmbilleder til ordre siden, indkøbskurv og betalingssiden på hjemmesiden.
- Lave koden til at lægge en vare i indkøbskurven og sende den videre til betaling.

- En ordre skal tilknyttes en kunde.
Da vi ikke har oprettet nogle nye vare endnu vil der blive lavet nogle fiktive vare.

Sprint 3 14 dage.
UC: # 3.1 Opret ordre (kunde)
Produkt backlog:
- Vi skal lave kode til opret ordre.
- Vi skal lave kode til betaling af vare.
- Der skal laves en tabel i databasen til ordre som skal tilknyttes kundetabellen.

Sprint 4 14 dage.
UC: #5.1 Opret varegruppe
Produkt backlog:
- Der skal laves skærmbilleder til hver varegruppe på hjemmesiden.
- Der skal laves skærmbilleder til vores salgssystem med varegrupper.
- Der skal oprettes en tabel til varegrupper i databasen.
- Vi skal kode så det er muligt at oprette en varegruppe for salgschefen.
- Opdatering af designklassediagram

UC: #5.2 Søg og udskriv varegruppe
Produkt backlog:
- Vi skal kode så det er mulig at søge efter varegruppe ud fra id eller navn.
- Den søgte varegruppe skal også være muligt at udskrive.

UC: #5.3 Opdatere varegruppe
Produkt backlog:
- Der skal kodes til Opdatere varegruppe.

UC: #5.4 Slet varegruppe
Produkt backlog:
- Der skal kodes til at kunne slette en varegruppe og den skal tjekke på om varegruppen er tom inden det kan ske.

Sprint 5 14 dage.
UC: #4.1 Opret vare
Produkt backlog:
- Der skal laves skærmbilleder til hver vare på hjemmesiden.
- Der skal laves skærmbilleder til vores salgssystem med vare.
- Der skal laves en tabel i databasen til vare som skal tilknyttes varegrupper.
- Det skal være muligt for salgschefen at oprette en ny vare som kommer op på hjemmesiden og bliver gemt i databasen.
- Hvis en vare når et minimumsantal, skal systemet sende en mail til salgschefen.

UC: #4.2 Søg vare
Produkt backlog:
- Der skal kodes så det er muligt at søge en vare ud fra id, navn og leverandør.

UC: #4.3 Opdatere vare
Produkt backlog:
- Der skal kodes så det er muligt at opdatere en vare.

UC: #4.4 Slet vare
Produkt backlog:
- Der skal kodes så det er muligt at slette en vare.

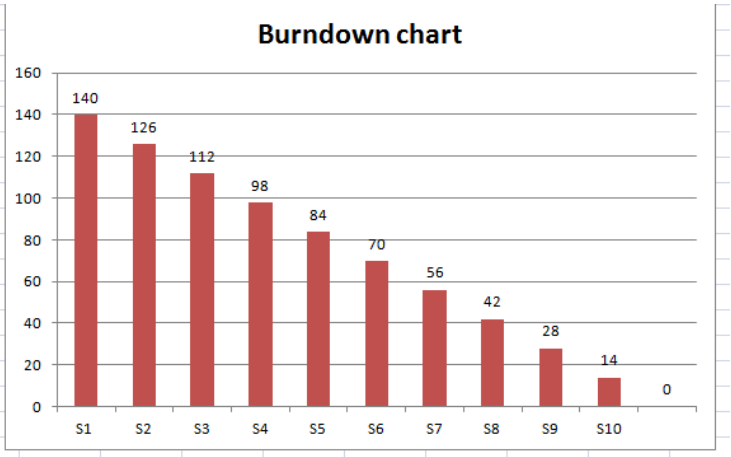
Sprint 6 **14 dage.**
UC: # 7.1 Opret ordre (medarbejder)
Produkt backlog:
- Skærmbilleder til ordre siden og betalingssiden i vores salgssystem.
- Det skal kodes så det er muligt, at oprette en ordre for en medarbejder.
UC:#3.2 Søg ordre(kunde)
Produkt backlog:
- Der skal kodes så kunden kan finde en ordre på hjemmesiden ved at søge ud fra .....
UC: #7.2 Søg ordre(medarbejder)
Produkt backlog:
- Der skal kodes så medarbejderen i salgs systemet kan finde en ordre ved at søge ud fra .....
UC: #3.3 Opdatere ordre(kunde)
Produkt backlog:
- Der skal kodes så kunde kan opdatere en ordre, hvis denne ordre ikke er afsendt endnu.
- Der skal kodes så hjemmesiden kan beregne for kundens ordre, hvis kundens opdatering koster mere eller mindre.

Sprint 7 **14 dage.**
UC: #7.3 Opdatere ordre(medarbejder)
Produkt backlog:
- Opdatering af designklassediagram.
- Der skal kodes så en medarbejder kan opdatere en ordre, hvis denne ordre ikke er sendt afsted endnu.
UC:#3.4 Slet ordre(kunde)
Produkt backlog:
- Der skal kodes så det er muligt for en kunde at slette ordren på hjemmesiden, hvis den ikke er sendt afsted endnu.

UC:#7.4 Slet ordre(medarbejder)
Produkt backlog:
- Der skal kodes så det er muligt for en medarbejder at slette ordren i salgs systemet, hvis den ikke er sendt afsted endnu.
UC: #1 Søg på hjemmesiden
Produkt backlog:
- Der skal kodes så det er muligt at søge efter vare på hjemmesiden ved hjælp af id, navn og leverandør.

Burndown chart(Lasse)

			140
S1	14	dage	126
S2	14	dage	112
S3	14	dage	98
S4	14	dage	84
S5	14	dage	70
S6	14	dage	56
S7	14	dage	42
S8	14	dage	28
S9	14	dage	14
S10	14	dage	0
I alt	140	dage	



Vores sprints kommer til at foregå over 140 dage for vores salgsmodul. Vi kører i 14 dages sprint, så vi hele tiden har kunden med ind over systemet og kunden hele tiden får leveret små software dele. Vi har forsøgt at estimere vores use cases og hvad der skal laves i dem (vores produkt backlog). Vi har estimeret hver use case og givet dem en tidsramme for, hvor lang tid vi tror det tager at lave dem. Så har vi taget de use cases som vi tror der vil give mest værdi for kunden og sat dem i rækkefølge efter det.

Test strategi (Tobias)

Udarbejdelse af afsnittet om test strategi har vi brugt Software Engineering, A Practioner's Approach 7. edition, kapitel 17 og 18.

Test er et vigtigt element i ethvert software projekt, da et program ikke er færdigt før det er fejlfrit, brugervenlige og klar til brug. Et program er fejlfrit indtil næste fejl er fundet! Betyder det så at man aldrig er færdig med at teste? Ikke nødvendigvis. Når programmet udgives er det brugerne der overtager test af softwaren, men indtil da, så er det software teamet der tester. Det behøver ikke nødvendigvis at være to forskellige personer der programmerer og tester softwaren. "Mindre" tests som unit- og integrationstest kan sagtens udføres af den person der har programmeret modulerne.

Følgende er en definition af relevante testmetoder samt en teststrategi for salgsmodulet.

Unit test

Unit testning er typisk den første test der vil forekomme. I unit tests tester man de mindste dele af systemet, de enkelte delelementer, som blandt andet kan være simple CRUD kommandoer. Unit testning koncentrerer sig om testning af logikken og datastrømmen inden for en komponent.

Når man unit tester ser man på at datastrømme ind i modulet er korrekte og at de ikke ændrer sig til uventede værdier eller typer. Samtidig testes om det er muligt, at nå hvert eneste scenarie i if-else statements og om loops holder sig indenfor grænserne. Loops, arrays og lignende mekanismer fejler oftest ved grænserne af det definerede spektrum. Hvad sker der når arrayet kommer til det n'te element og loopet passere tællerens værdi?

Til sidst tester man ekstensivt de fejlscenarier en kunde kan finde på at kreere, selvom det er svært at forestille sig hvad de kan finde på. Selv hvis fejl opstår, og det gør de højst sandsynligt på et tidspunkt, så er det nødvendigt at designe moduler således, at fejlmeddelelserne bliver forståelige for brugerne og modulet viderestiller til næste vindue. Fejlmeddelelser skal ikke sige "Du burde ikke kunne få denne fejl", da det 1. ikke hjælper brugeren videre og 2. fortæller brugeren at noget er alvorligt galt.

Integrationstest

Integrationstest udføres for at finde ud af om komponenterne man har unit testet kan interagere med hinanden. Integrationstests ser på hvilke værdier, og hvis programmeringssproget er typestærkt, hvilke typer der bliver returneret eller sendt videre. Stemmer værdierne og typerne ikke overens med de metoder der skal modtage får man problemer. Samtidig kan data blive tabt på tværs af interfaces, eksempelvis mellem salgsmodulet og serveren, eller databasen.

Strategi

I forbindelse med kodningen af salgsmodulet ønsker vi at unit teste umiddelbart imens eller efter at kodningen af de enkelte delelementer er færdige. Unittesten udføres af personen, der har programmeret delelementet, fordi han har størst kendskab til dets kald, typer og værdier.

Testmiljø: Testen skal udføres på seneste version af Visual Studio i et Windows 8.1 miljø, da det højst sandsynligt vil være det kunden kommer til at køre systemet på.

Test grupper: Integrationstesten af delelementerne i salgsmodulet vil blive udført som par test af teammedlemmerne. Her vil det blive relevant at teste forbindelse mellem klasser, webshop og databasen.

Test Krav: Både unit- og integrationstesten skal udføres manuelt. Til hver test skal der stilles klare forventninger om udfaldet. Først når disse forventninger er indfriet eller dokumenteret opfattes testen som færdig.

Test tidsplan: Vi forventer at selve testningen vil foregå i samme sprint som kodningen tager sted og som projektet skrider frem vil der også indgå integrationstest mellem de enkelte delelementer løbende.

Test resumé: Det forventes at der tages noter og væsentlige fejl, ændringer og bemærkninger skrives i et dokument sideløbende med test. Dokumentet skal afleveres sammen med salgsmodulet.

Konklusion (Alle)

Hvor det har været tanken at vi skulle arbejde efter en SCRUM-procesmodel fra start til slut, har vi dog ikke arbejdet efter denne procesmodel i det første inkrement. SCRUM modellen ville have at vi skulle arbejde i sprint fra start til slut, dette har ikke været tilfældet, grundet en præference i at have et større fundament, hvorpå vi kan arbejde videre fra. Fundamentet i dette tilfælde er et objekt orienteret analyse fase, som indeholder elementer fra systemudviklingsfagene, fra første og frem til fjerde semester på datamatiker uddannelsen i vejle. I det fortsatte forløb vil vi arbejde mere iterativt end vi har gjort under dette forløb. Herved sikrer vi en større succes for projektet, grundet vi har kunden med under hele vores udvikling. Kunden kan derved være med til at styre projektet, så de ender med et produkt som de kan blive tilfreds med.

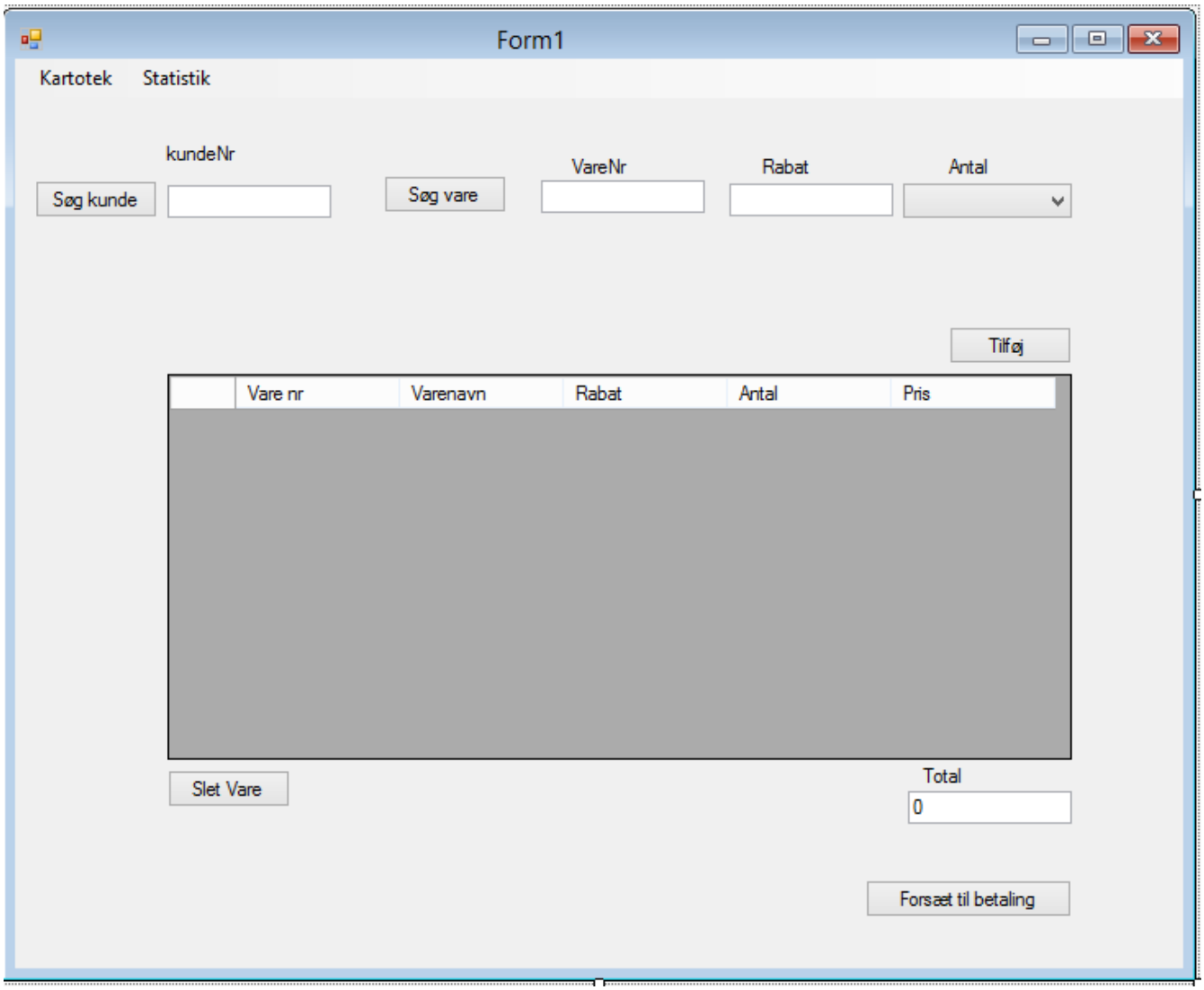
Bilag

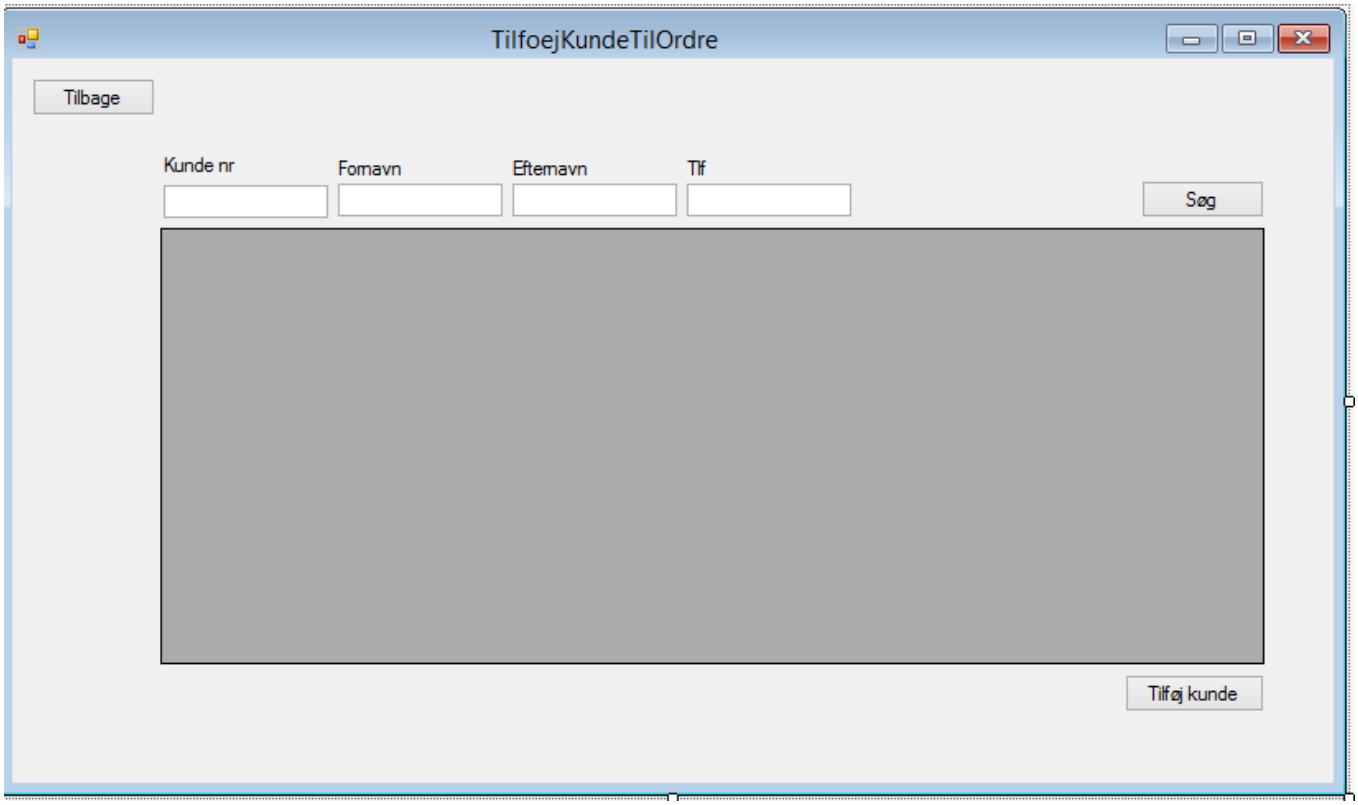
Lasse

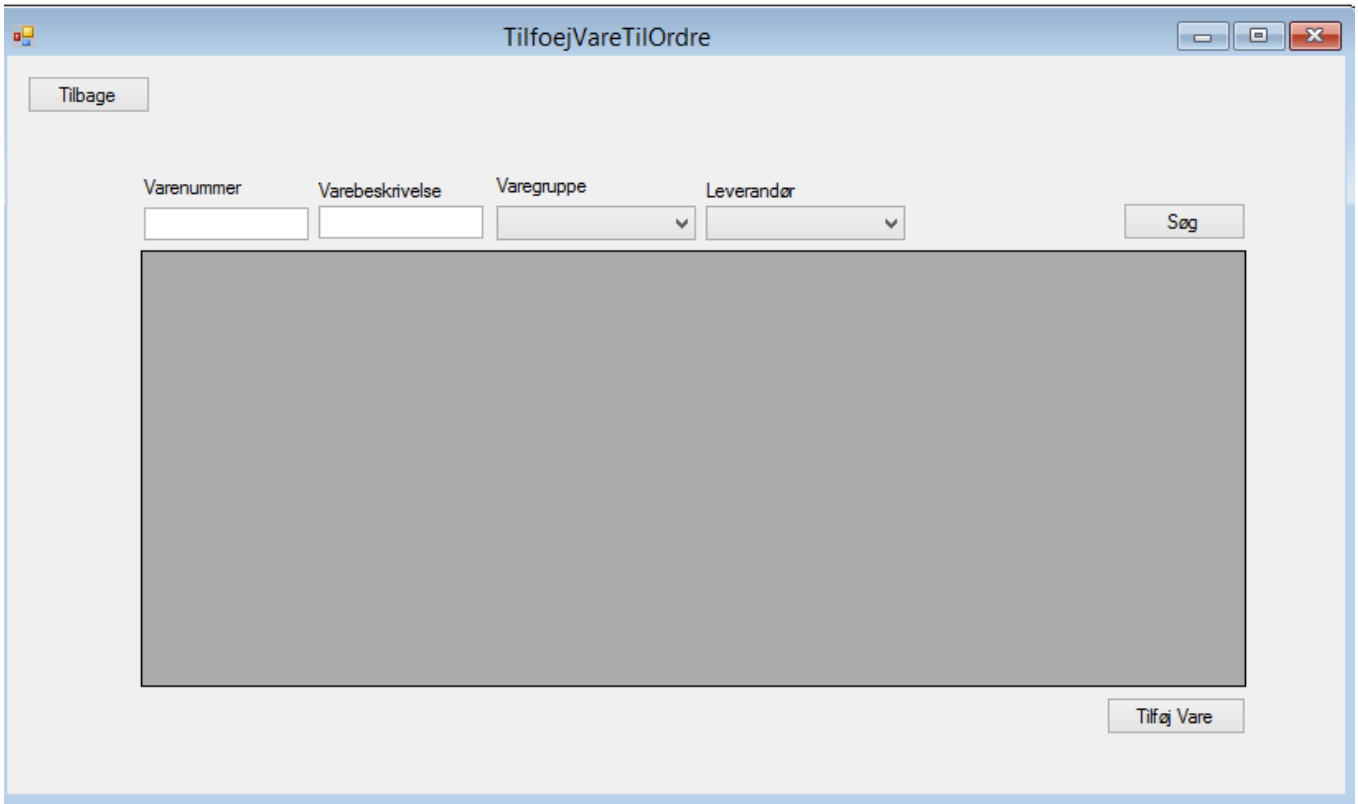
Skærmbilleder

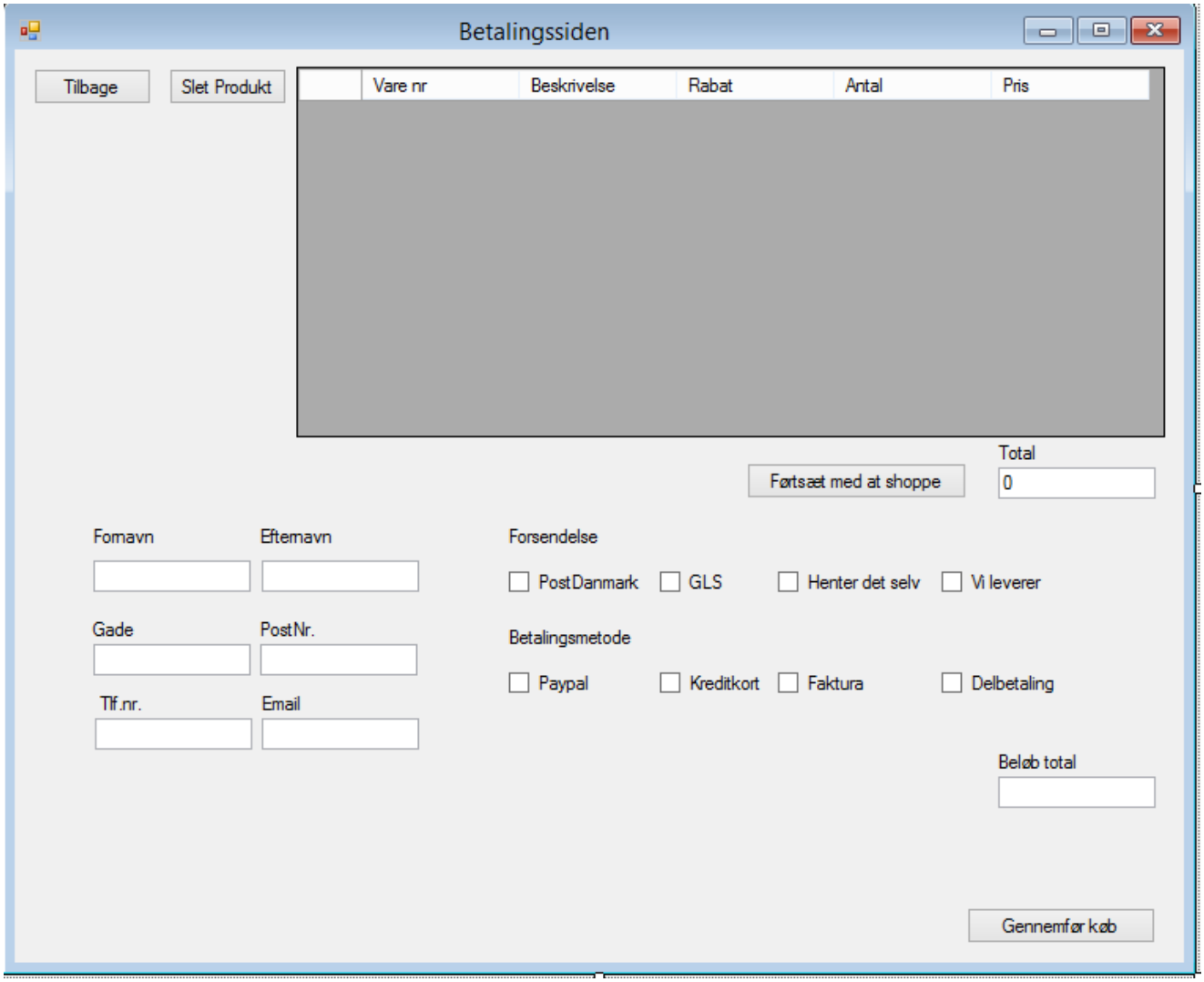
ERP-salgs systemet(1.1)

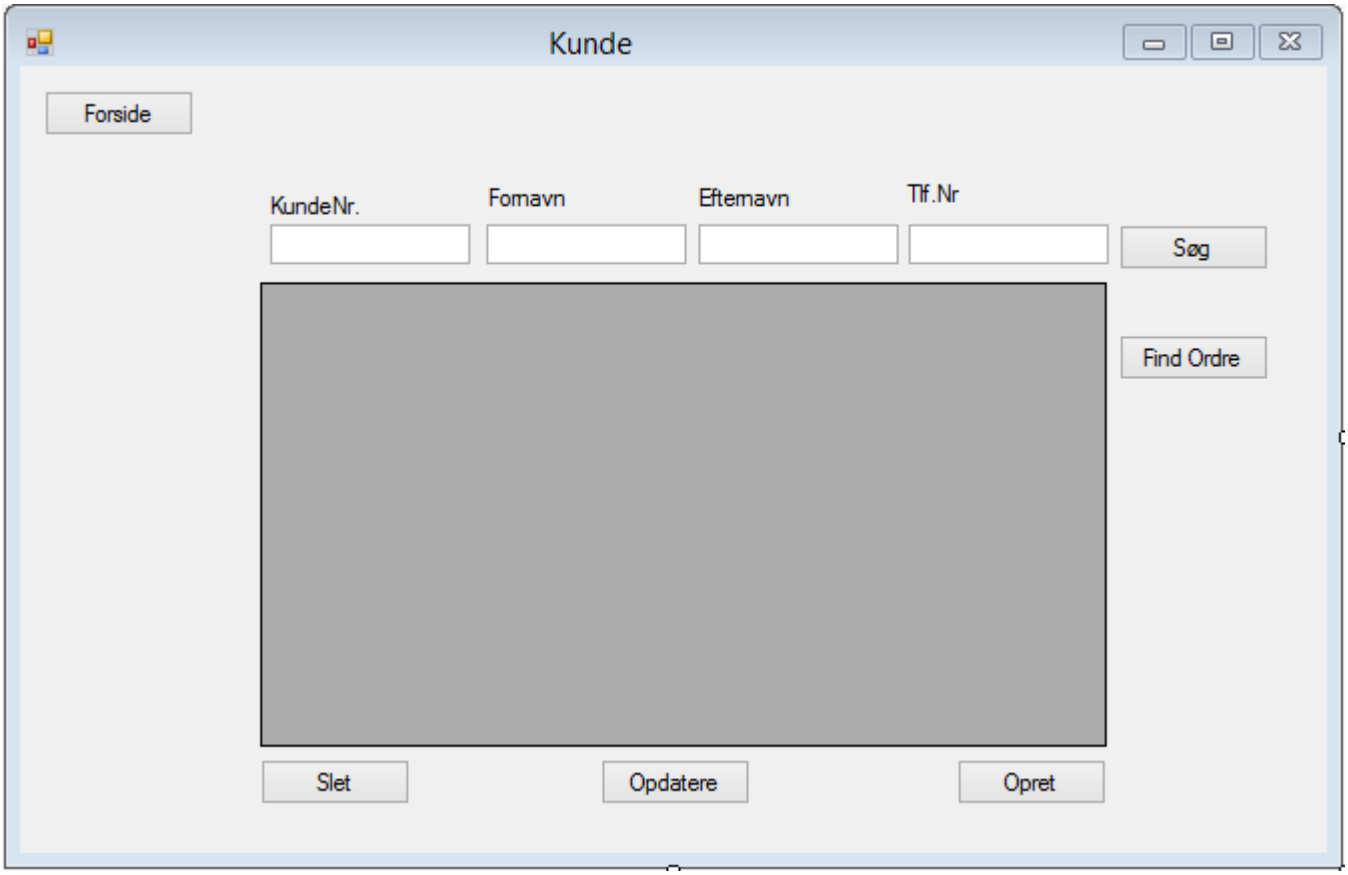
Forsiden

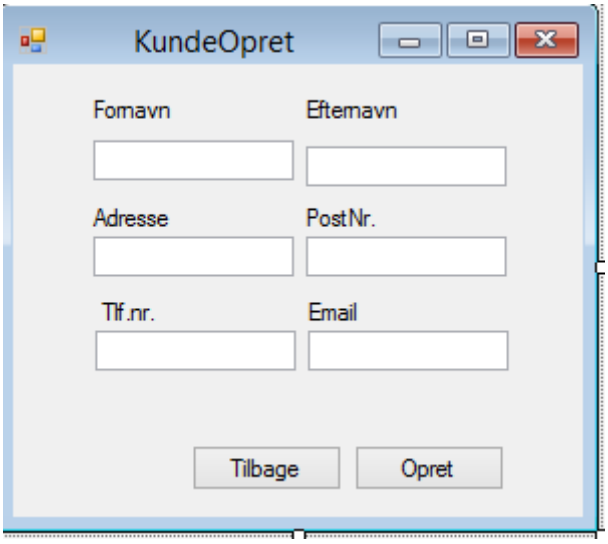



TilfoejKundeTilOrdre
Tilbage
Kunde nr
Fornavn
Efternavn
Tlf
Søg
Tilføj kunde

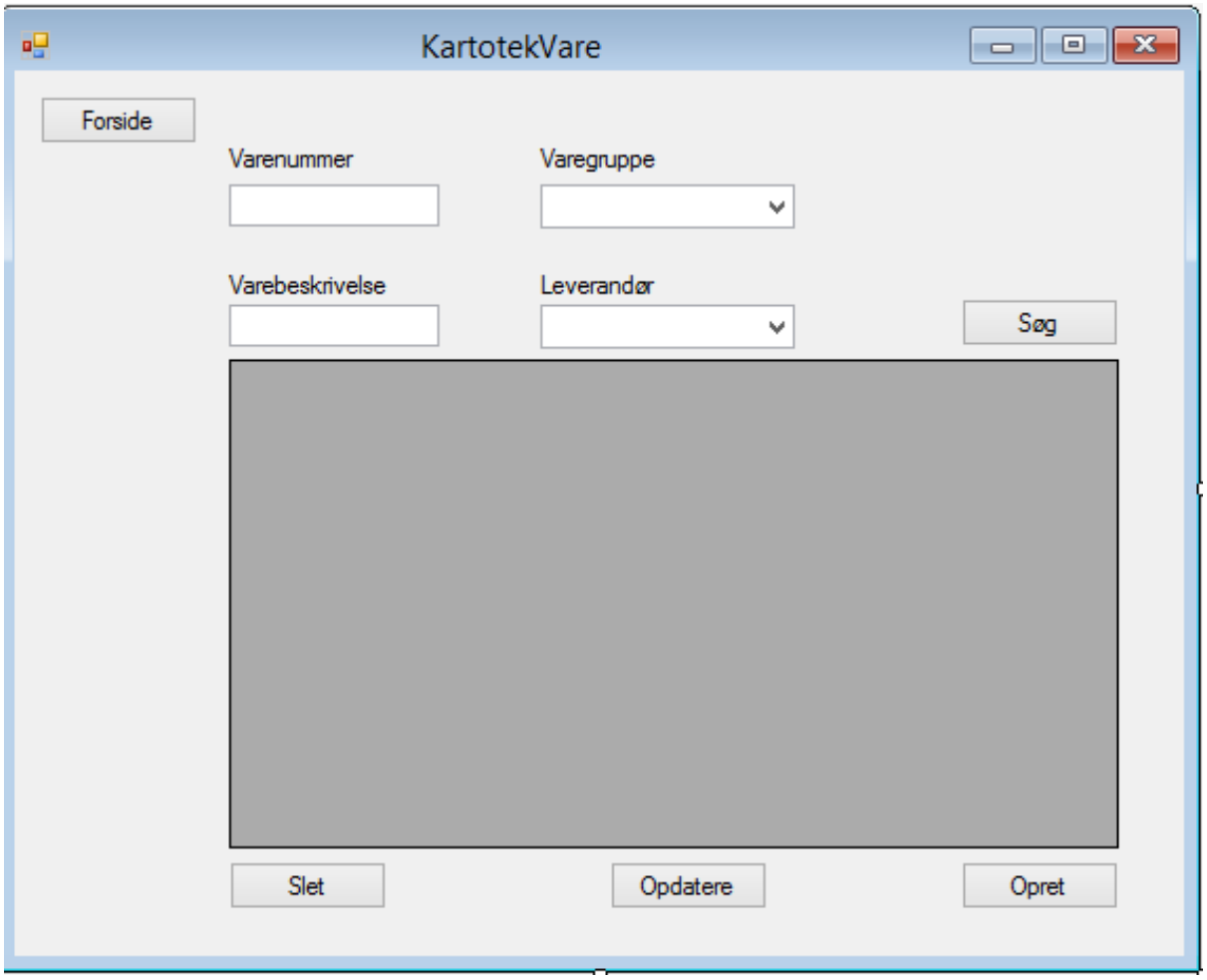

TilfoejVareTilOrdre
Tilbage
Varenummer
Varebeskrivelse
Varegruppe
Leverandør
Søg
Tilføj Vare

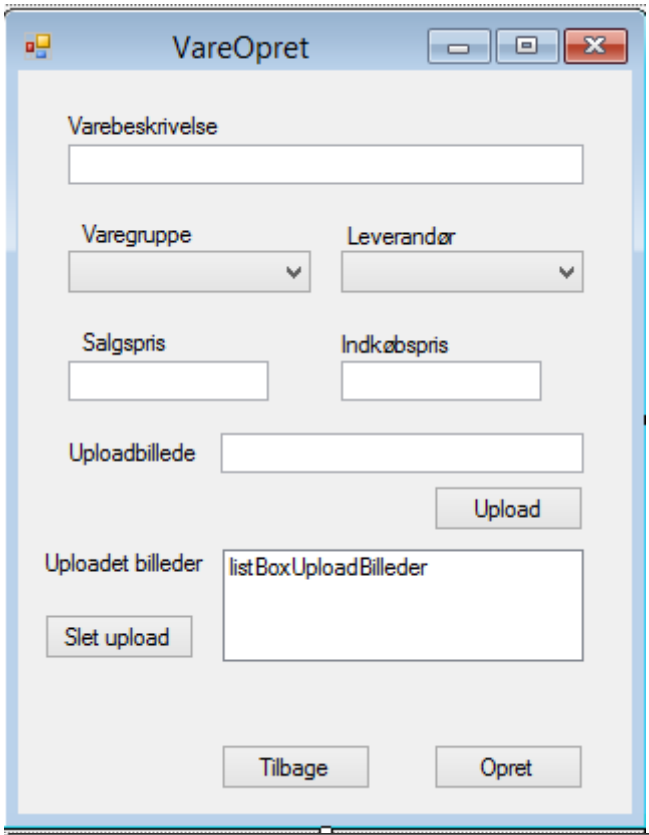

Betalingssiden
Tilbage
Slet Produkt
Vare nr
Beskrivelse
Rabat
Antal
Pris
Total
0
Førtsæt med at shoppe
Fornavn
Efternavn
Gade
PostNr.
Tlf.nr.
Email
Forsendelse
PostDanmark
GLS
Henter det selv
Vi leverer
Betalingsmetode
Paypal
Kreditkort
Faktura
Delbetaling
Beløb total
Gennemfør køb

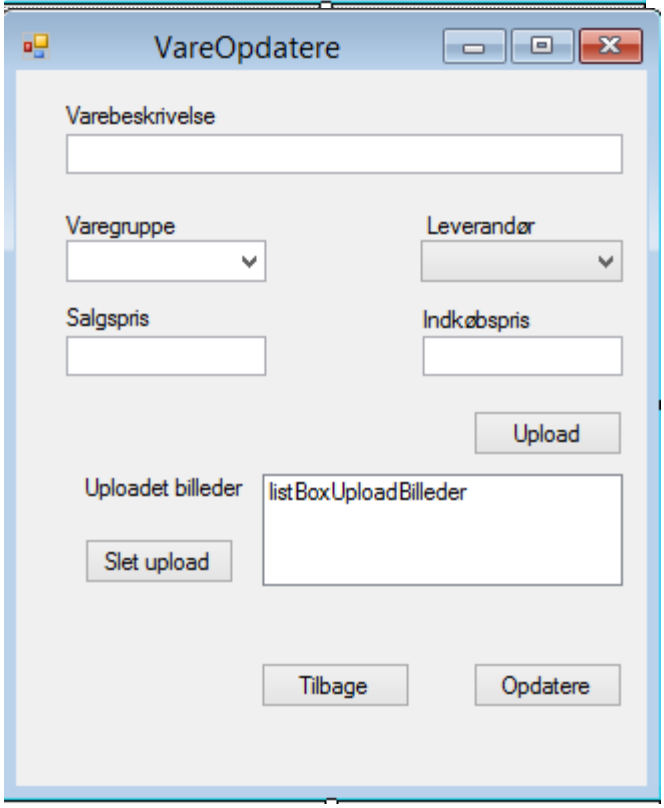

Kunde
Forside
KundeNr.
Fornavn
Efternavn
Tlf.Nr
Søg
Find Ordre
Slet
Opdatere
Opret


KundeOpret
Fornavn
Efternavn
Adresse
PostNr.
Tlf.nr.
Email
Tilbage
Opret

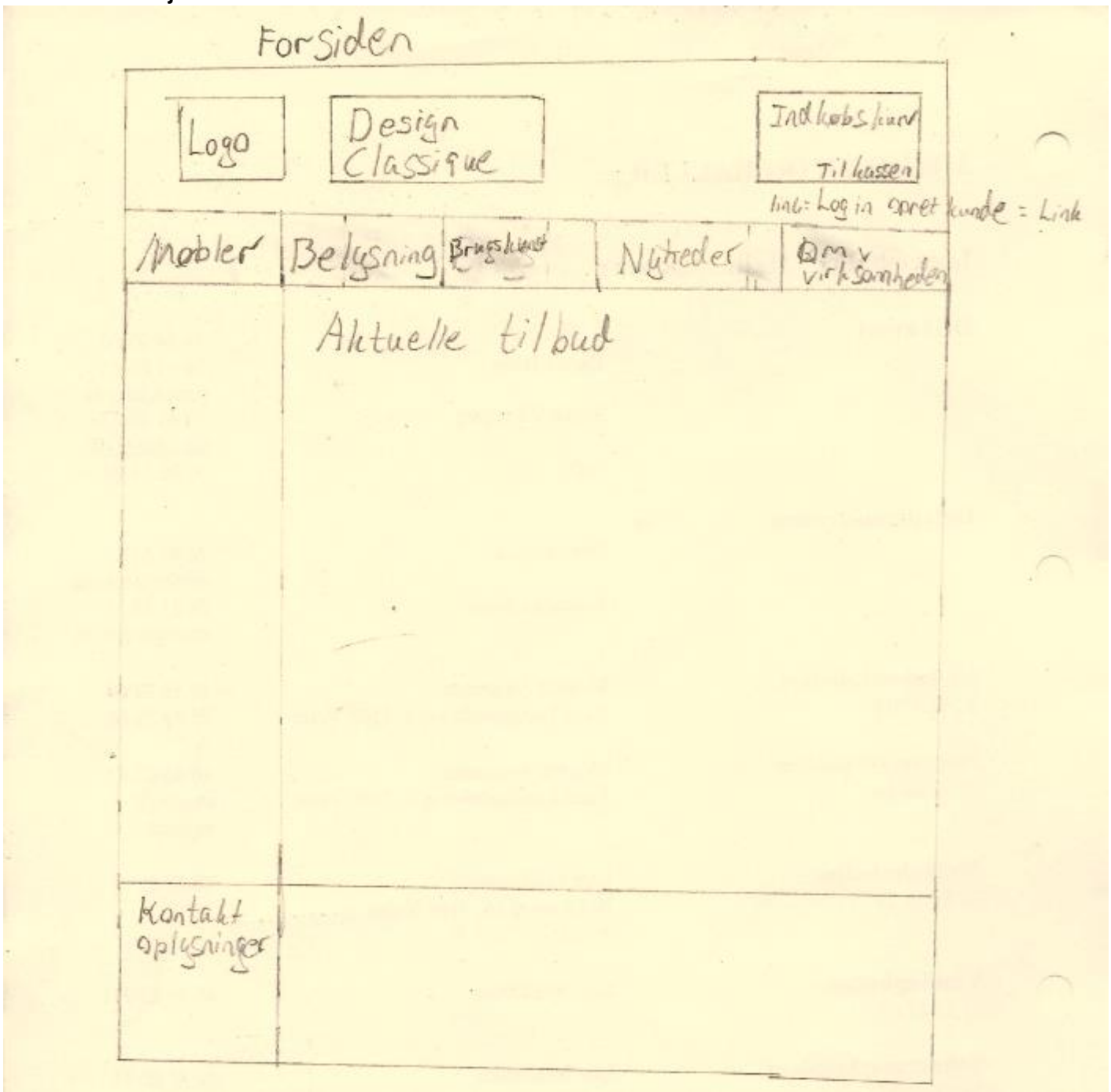

KundeOpdatere
Fornavn
Efternavn
Gade
PostNr.
Tlf.nr.
Email
Tilbage
Opdatere

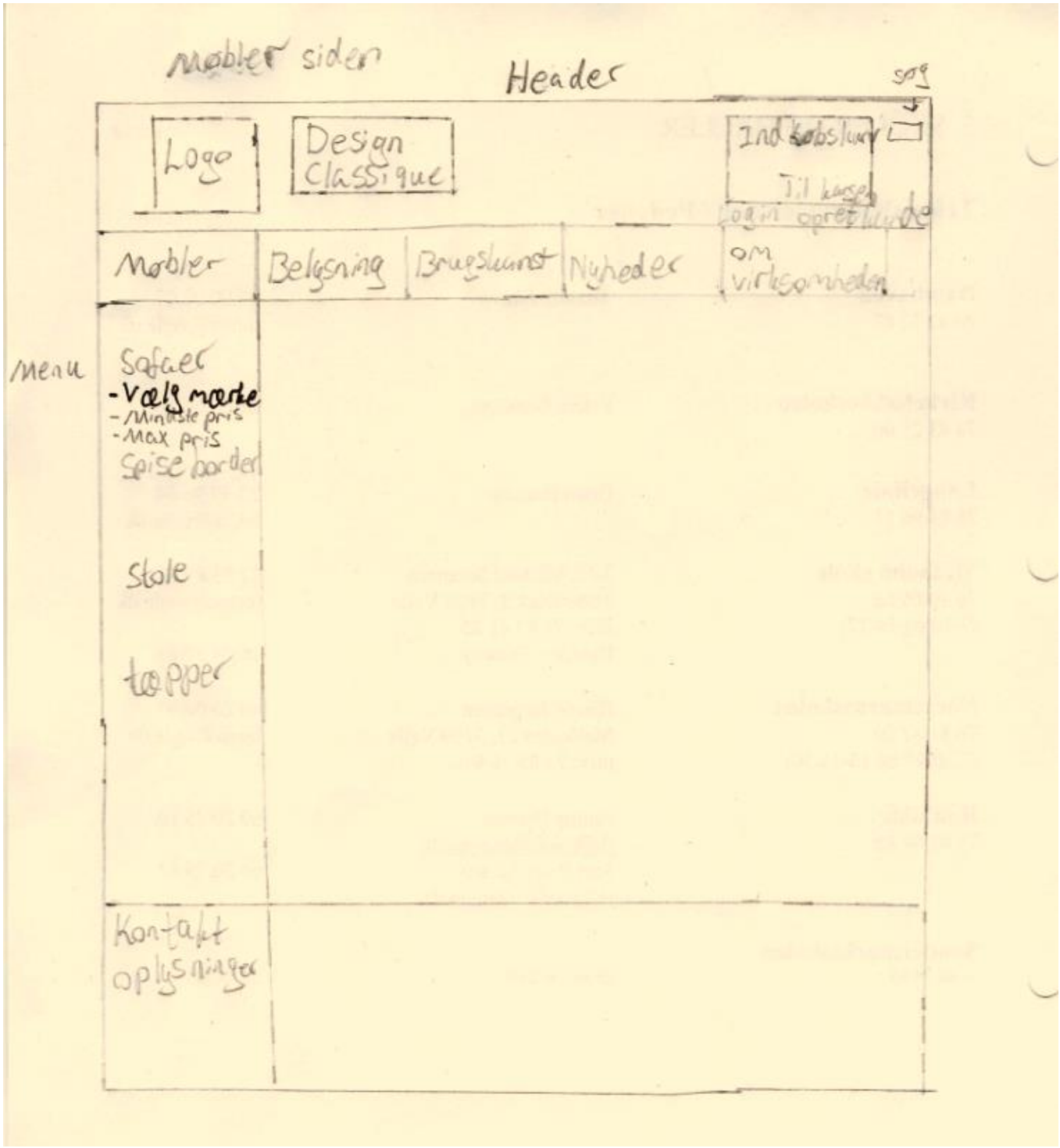

KartotekVare
Forside
Varenummer
Varegruppe
Varebeskrivelse
Leverandør
Søg
Slet
Opdatere
Opret

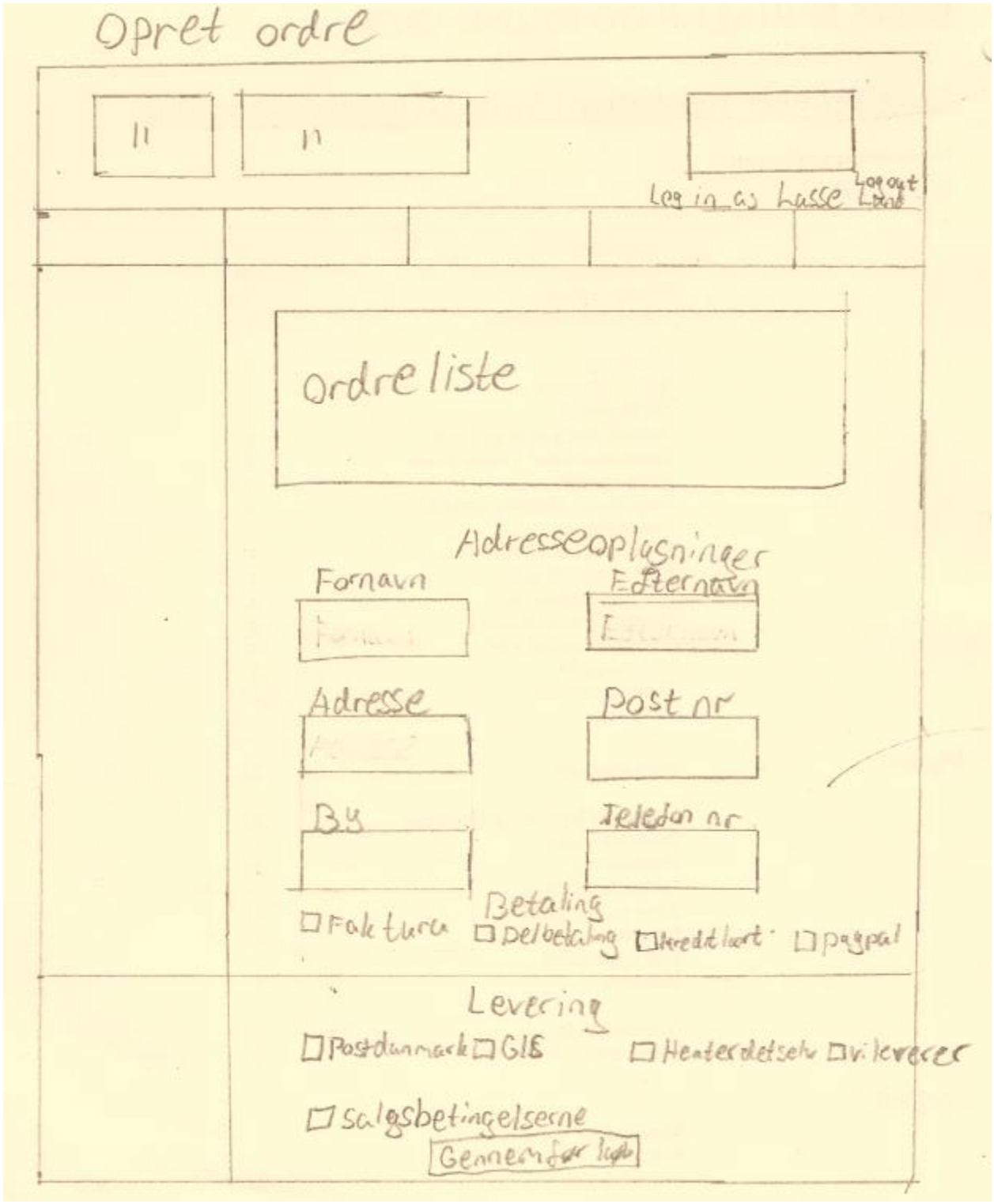

VareOpret
Varebeskrivelse
Varegruppe
Leverandør
Salgspris
Indkøbspris
Uploadbillede
Upload
Uploadet billeder
listBoxUploadBilleder
Slet upload
Tilbage
Opret


VareOpdatere
Varebeskrivelse
Varegruppe
Leverandør
Salgspris
Indkøbspris
Upload
Uploadet billeder
listBoxUploadBilleder
Slet upload
Tilbage
Opdatere

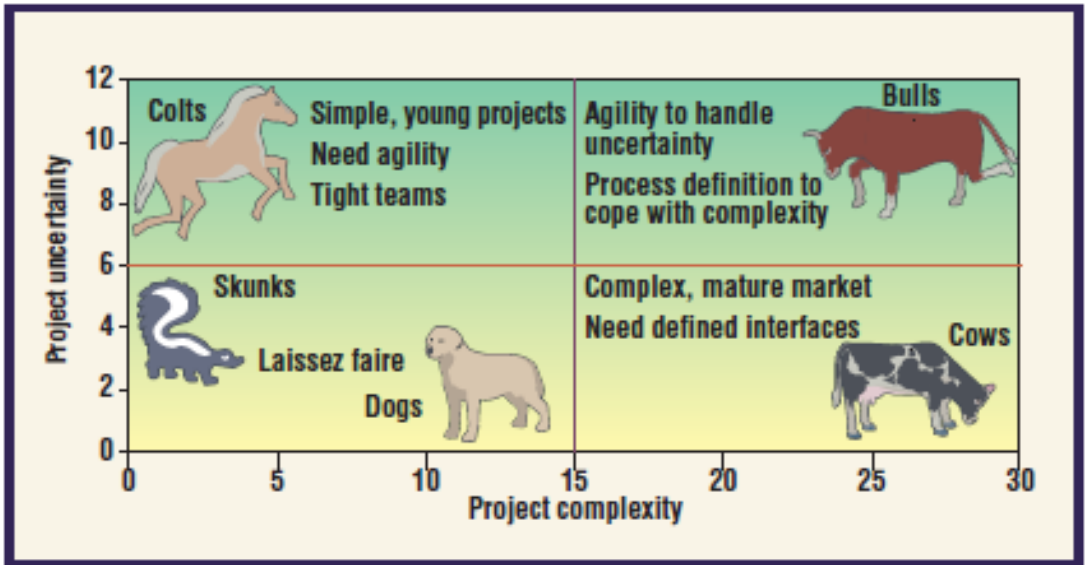
Bilag(1,2)
Skærmbilleder hjemmesiden.




møbler siden
Header
søg
Logo
Design Classique
Ind
Til kasse
Møbler
Belysning
Brugskunst
Nyheder
Om virksomheden
Menu
Sofaer
-Vælg mærke
-Mindste pris
-Max pris
Spisebord
Stole
toppe
Kontakt oplysninger


Opret ordre
Log in as Lasse
Log out
ordre liste
Adresseoplysninger
Fornavn
Efternavn
Adresse
Post nr
By
Telefon nr
Betaling
Faktura
Delbetaling
Kreditkort
paypal
Levering
Postdanmark
GLS
Henterdetselv
vi leverer
Salgsbetingelserne
Gennemfør køb

Todd Little complexity and uncertainty



Resterne sprints

Sprint 8 **14 dage.**

UC: #2.2 Søg kundeprofil

Produkt backlog:

- Der skal kodes så det er muligt at finde en kundeprofil ud fra id, navn og email.

UC: #2.3 Opdatere kundeprofil

Produkt backlog:

- Der skal kodes så det er muligt på hjemmesiden for kunden at opdatere oplysninger.

UC: #2.4. Slet kundeprofil

Produkt backlog:

- Der skal kodes så kunden kan slette sin profil.

UC: #8.1 Opret medarbejder

Produkt backlog:

- Der skal laves skærmbilleder for medarbejder i salgssystemet.
- Der skal kodes så slagschefen kan oprette en medarbejder.
- Der skal laves en medarbejder tabel i databasen som skal tilknyttes .....

UC: #8.2 Søg medarbejder

Produkt backlog:

- Der skal kodes så salgschefen kan finde en medarbejder ud for id eller navn.

UC: #8.3 Opdatere medarbejder

Produkt backlog:

- Der skal kodes så salgschefen kan opdatere en medarbejder.

UC: #8.4 Slet medarbejder
Produkt backlog:
- Der skal kodes så salgschefen kan slette en medarbejder.

Sprint 9 14 dage.
UC: #6.1 Opret kundeprofil (medarbejder)
Produkt backlog:
- Der skal laves skærmbilleder for kundeprofil i salgssystemet.
- Der skal kodes så en medarbejder kan oprette en kunde i salgssystemet.

UC: #6.2 Søg kundeprofil (medarbejder)
Produkt backlog:
- Der skal kodes så det er muligt at finde en kundeprofil ud fra id, navn og email.

UC: #6.3 Opdatere kundeprofil(medarbejder)
Produkt backlog:
- Der skal kodes så det er muligt i salgssystemet for kunden at opdatere oplysninger.

UC: #6.4 Slet kundeprofil(medarbejder)
Produkt backlog:
Der skal kodes så en medarbejder kan slette en kundeprofil.

Sprint 10 14 dage.
UC: #9.1 Opret leverandør
Produkt backlog:
- Der skal laves skærmbilleder for leverandør i salgssystemet.
- Der skal laves en leverandør tabel i databasen som er tilknyttet vare.
- Der skal kodes så det er muligt for salgschefen at oprette en leverandør.

UC: #9.2 Søg leverandør
Produkt backlog:
- Der skal kodes så det er muligt for salgschefen at finde en leverandør ud fra id eller navn.

UC: #9.3 Opdatere leverandør
Produkt backlog:
- Der skal kodes så det er muligt for salgschefen at opdatere en leverandør.

UC: #9.4 Slet leverandør
Produkt backlog:
- Der skal kodes så det er muligt for salgschefen at slette en leverandør.

Lars

Resterende RMMM planer:

Risiko informationsskema			
Risiko ID: ID_03	Dato: 04/05-14	Sandsynlighed 10 %	Konsekvens 1
Beskrivelse: Manglende systemdata grundet backup failure. Dette kan skyldes manglende opfølgning på backup eller at programmet, der anvendes som backup, svigter.			
Mitigation: Konsekvensen for denne risiko er blevet estimeret til 1. Hvis risikoen indtræffer, kan det få katastrofale følger for projektet. Følgerne består i at vi må snakke med kunden om, hvad vi skal gøre. Skal vi arbejde videre med det vi har eller skal projektet helt stoppe. Vi risikerer altså at fejle opgaven. **Monitoring:** Når vi arbejder med backup er det vigtigt, at have aftaler med vores backup-firma, hvis der findes et. Det er også vigtigt at holde øje med de backups vi laver ikke er redundante. Vi skal også have fokus på at tage daglig backup for ikke at risikerer et for stort tab af data.			
Management/contingency plan/trigger: Skulle risikoen indtræffe må vi snakke med kunden om vi skal fortsætte med data vi måtte have. Er der et for stort data til at vi kan arbejde videre og kunden ikke er tilfreds med hvad han/hun har fået, må vi opgive projektet og betale de penge tilbage som kunden måtte have betalt.			
Initiativtager: Lars Skaaning Jensen		Underskrevet: Lars Skaaning Jensen	

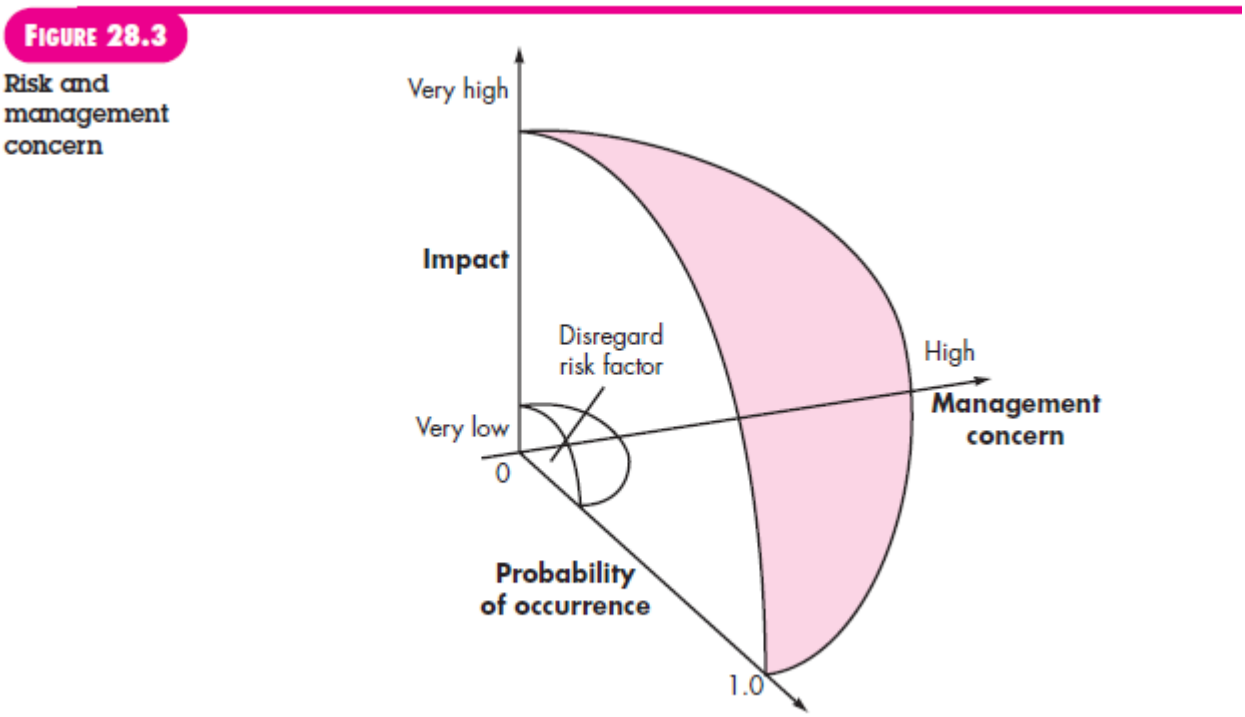
ID_04 er ikke skrevet fordi den ikke er relevant at regne på eller holde øje med, så længe vi har møder med kunden og ikke fortsætter projektet uden kunden er tilfreds.

Figur Lars1:

Components / Category		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

1 viser den overordnede betydning for fejlen i projektet.
2 betyder i figuren hvad fejlen kan være.

Figur Lars2:



Christopher

Følgende er resterende use cases:

Use case #1 søg på hjemmesiden

Primær aktør: Kunde

Pre-condition: pc/mac med browser

Kunde	Hjemmeside
1. Kunde åbner webbrowser	
2. Kunde går ind på webadresse	
	3. Hjemmeside vises
4. Kunde benytter søgefunktion	
	5. hjemmeside viser resultat af søgning

Post condition: Kunde ser resultat af søgning på hjemmesiden.

Use case #2 CRUD kunde profil

Primær aktør: Kunde

Use case #2.2 Læs kundeprofil. Post condition: Kundeprofilen vises.

Use case #2.4 Slet kundeprofil. Post condition: Kundeprofilen er slettet.

Use case #3 CRUD kundeordre

Use case #3.2 Søg ordre. Post condition: Ordre fundet.

Use case #3.3 Opdater ordre. Post condition: Ordre er opdateret.

Use case #3.4 Slet ordre. Post condition: Ordre er slettet.

Use case #4 CRUD vare

Use case #4.2 Søg vare. Post condition: Vare er fundet.

Use case #4.3 Opdater vare

Primær aktør: Salgschef

Pre-condition: Salgschefen er logget ind i ERP - systemet.

Salgschef	System
1. Salgschef åbner varekartotek	
	2. System viser varekartotek
3. Salgschef søger efter vare	
	4. System viser resultat af søgning
5. Salgschef vælger at ændre vareoplysninger	
6. Salgschef gemmer ændring	

Post condition: vare opdateret.

Use case #4.4 Slet vare

Primær aktør: Salgschef

Pre-condition: Salgschefen er logget ind i ERP - systemet.

Salgschef	System
1. Salgschef markerer vare	
2. Salgschef vælget at slette vare	
	3. System beder salgschef bekræfte valg af sletning
4. Salgschef vælger at bekræfte, at han vil slette vare(r)	
	5. System bekræfter sletning

Post condition: Vare er slettet fra systemet.

Use case #5 CRUD varegruppe

Primær aktør: Salgschef.

Use case #5.1 Opret varegruppe. Post condition: Varegruppe er oprettet.

Use case #5.2 Søg og udskriv varegruppe

Pre-condition: Salgschefen er logget ind i ERP - systemet.

Salgschef	System
1. Salgschef åbner varegruppe info	
	2. System viser varegruppes info og lagerstatus
3. Salgschefen vælger at udskrive varegruppe.	
	4. Systemet udskriver valgte varegruppe.

Post condition: Salgschef har set lagerstatus/udskrevet varegruppe.

Use case #5.3 Opdater varegruppe. Post condition: Varegruppe er opdateret.

Use case #5.4 Slet varegruppe. Post condition: Varegruppe er slettet.

Use case #6 CRUD kundeprofil

Primær aktør: Medarbejder

Use case #6.1 Opret kundeprofil. Post condition: Kundeprofil oprettet.

Use case #6.2 Søg kundeprofil. Post condition: Kundeprofilen er fundet.

Use case #6.3 Opdater kundeprofil.

Pre-condition: Medarbejder er logget ind i ERP - systemet

Medarbejder	System
1. Medarbejder åbner kunde kartotek	
	2. System viser kunde kartotek
3. Medarbejder søger i kunde kartotek	
	4. System viser resultat af søgning
5. Medarbejder vælger at redigere valgte kundeprofil	
6. Medarbejder gemmer ændringer	
	7. System beder medarbejder om at bekræfte ændringer
8. Medarbejder bekræfter ændringer	
	9. System sender email angående redigering af profil til kunde

Post condition: Kundeprofil opdateret.

Use case #6.4 Slet kundeprofil. Post condition: Kundeprofilen er slettet.

Use case #7 CRUD kundeordre

Primær aktør: Medarbejder

Use case #7.1 Opret ordre for kunde

Pre-condition: Medarbejder er logget ind i systemet

Medarbejder	System
1. Medarbejder vælger opret ordre for kunde.	
	2. System viser vareliste
3. Medarbejder tilføjer vare til ordre.	
4. Medarbejder går videre til kunde oplysninger.	
	5. System viser kunde info side
6. Medarbejder udfylder kundes adresse info	
7. Medarbejder går videre til betalingsoplysninger	
	8. System viser betalings side
9. Medarbejder udfylder kundes betalingsoplysninger.	
10. Medarbejder foretager køb	
	11. System beder om bekræftelse af oplysninger
12. Medarbejder bekræfter oplysninger	
	13. System sender ordrebekræftelse til kunde

Post condition: Kundeordren er oprettet.

Use case #7.2 Søg kundeordre. Post condition: Kundeordre er fundet.

Use case #7.3 Opdater kundeordre

Primær aktør: Medarbejder

Pre-condition: Use case #6.1 step 1-4

Medarbejder	System
1. Medarbejder finder specific ordre	
2. Medarbejder vælger at redigere ordre	
3. Medarbejder gemmer ændring i ordre	
	4. System beder medarbejder om at bekræfte ændring
5. Medarbejder bekræfter ændring	
	6. System meddeler kunde per email og salgschef via system om ordreændring

Post condition: Kundeordre er opdateret.

Use case #7.4 Slet kundeordre. Post condition: Kundeordre er slettet.

Use case #8 CRUD medarbejder

Primær aktør: Salgschef

Use case #8.1 Opret medarbejder

Use case #8.2 Søg medarbejder

Use case #8.3 Opdater medarbejder

Use case #8.4 Slet medarbejder

Use case #9 CRUD leverandør

Primær aktør: Salgschef

Use case #9.1 Opret leverandør

Use case #9.2 Søg leverandør

Use case #9.3 Opdater leverandør

Use case #9.4 Slet leverandør