Vestbjerg Byggecenter A/S

Paw Gilkrog

Thais Gilkrog

Marc Rene Jensen

Jacob Winther Nielsen

Vejledere:
Lise Klitsgaard
Gunhild Andersen
Henrik Hvarregaard
Anita Clemmensen

Abstract
This paper is about the company Vestbjerg Byggecenter A/S, who wishes a new it-system to manage their business, which will help them compete on the market. Their old system is outdated and not fit for a company of their size. This paper will analyse the company, and program a system based on the needs of the company.

Proffesionshøjskolen

University College Nordjylland

Teknologi og Business: IT Uddannelserne

December 14, 2016

1. Forord

Gruppen har fået en opgave som går ud på, at der skal lave et nyt system til virksomheden Vestbjerg Byggecenter A/S, eftersom deres gamle UNIX-system er forældet og ikke i stand til at gøre det, som firmaet ønsker og har brug for. Der blev udleveret en opgavetekst som gruppens medlemmer læste igennem, og har derefter arbejdet ud fra de oplysninger, som teksten har givet. Bl.a. skal det nye system kunne bogføre ordrene, så medarbejderne ikke er nødt til at sidde i forskellige regneark og holde styr på diverse ting via. disse.

TortoiseSVN revisions = 122
https://kraka.ucn.dk/svn/dmaa0916_1$Sem_Project_8$

1. Contents

2. Indledning

Virksomheden Vestbjerg Byggecenter A/S ønsker at skifte deres ældre Unix it-system ud med et nyere system, som skal være med til at forbedre virksomhedens konkurrence evne. Gruppen vil starte med en it-forundersøgelse af virksomheden og ud fra denne forundersøgelse, og en opsummering med Parker/Benson Matrice, vil gruppen beslutte hvad der er i bedst interesse, for virksomheden at få skiftet deres UNIX-system ud med. Derefter vil der udføres system design, for den del af systemet som bliver lagt vægt på. Hvor der tilsidst vil blive beskrevet dele af koden i programmet, med en begrundelse.

3. It-forundersøgelse

I forbindelse med projektet er der blevet udarbejdet en it-forundersøgelse af virksomheden Vestbjerg Byggecenter A/S, hvori der laves en grundig analyse af virksomheden, baseret på vores viden fra virksomheds faget.

3.1 Analyse af nuværende situation

I dette afsnit bliver virksomhedens nuværende situation gennemarbejdet, med en analyse af virksomhedens struktur og organisationskultur, en tabel over problemer, hypoteser og løsningsmuligheder. Til sidst vil der være en vurdering af forskellige faktorer og en opsamling i SWOT.

3.2 Struktur

> *"Struktur er således ledelsens formelle værktøj. Det vil sige, at ledelsen beskriver, hvordan driften af organisationen skal fungere ved at opdele organisationen efter ansvars- og arbejdsfordeling. Dette kommer til udtryk gennem organisationsplaner, stillingsbeskrivelser og forretningsgange".* [2]

3.2.1 Organisationsprincip:

Vestbjerg Byggecenter A/S har Anders Olesen som ejer og direktør af virksomheden. Hans to sønner, Casper Olesen og Thomas Olesen, er begge ledere i hver sin afdeling. Casper er leder for trælasthandlen og Thomas er leder for byggemarkedet. Thomas har under sig en souschef og to ledere.

3.2.2 Arbejdsdelingsprincip:

Virksomheden anvender organiske principper, da man i virksomheden har etableret et pejse og brændeovnes center for nyligt. Hvilket vil sige, at i virksomheden er forandringshastigheden høj, og man er villig til at omstille sig selv.

3.2.3 Dybde (antal niveauer):

Der er tre niveauer i virksomheden. Niveau 0 er direktøren Anders Olesen. Niveau 1 er hans to sønner Casper og Thomas med deres afdelinger. Niveau 2 er køkken og bad afdeling, pejse og brændeovnes afdeling, samt kontor afdelingen.

3.2.4 Kontrolspænd (bredde):

I trælasthandelen har Casper Olesen 8 medarbejder under sig. I byggemarked har man i køkken/bad afdeling en leder og 2 sælger og i pejse/brændeovnes afdeling 1 leder og 1 ekspedient, derved 10 andre ansatte i byggemarkedet. Hvilket vil sige at kontrolspændet er begrænset på 10 medarbejder pr. leder, og organisationen hermed opfylder Fayol og den administrative skole på omkring 10 medarbejder.

3.2.5 Vurdering af organisationsstrukturen :

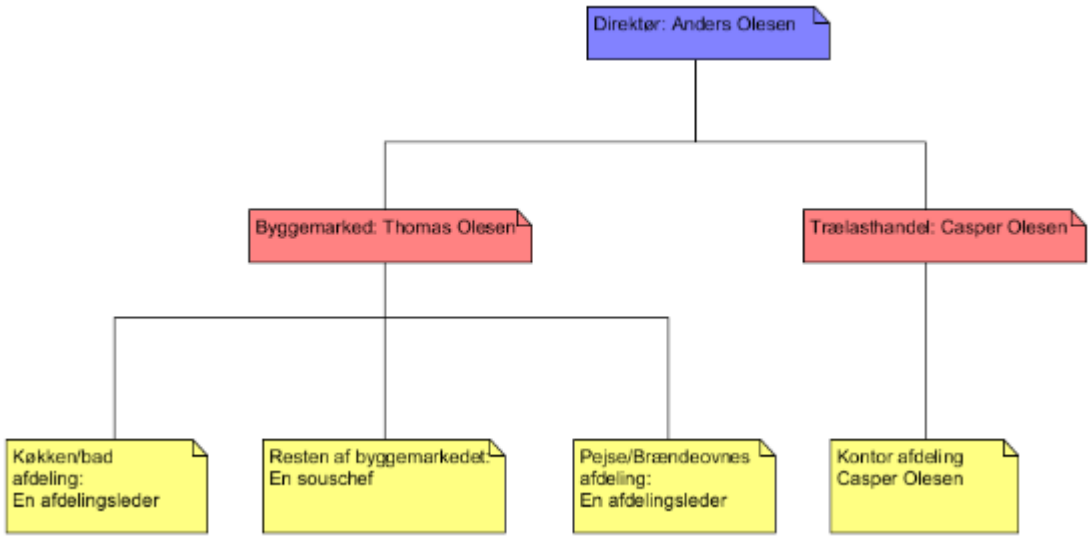


Figure 3.1: Organisation, opdelt i tre niveauer

Virksomheden er en organisk organisation, som forandrer sig med tiden. Der er en vertikal arbejdsdeling i organisationsstrukturen. Da man prøver at begrænse kontrolspændvidden ved at der er mere avanceret og komplekse opgaver, dermed bliver virksomheden en dyb organisation.

3.3 Organisationskultur

> *"En organisationskultur indeholder de værdier, normer, forståelse, holdninger, tro og tankemønstre, der deles af en organisations medlemmer".* [2]

3.3.1 Lederkarakteristik

For at lave en lederkarakteristik kigger man på en række elementer, som kan være, Hvad er en leder?: Hvor man kigger f.ek.s på ledergitter og lederstile. Hvem er lederen?: Hvor man kan kigge på lederroller og menneskesyn. Hvordan praktiseres ledelse?: Hvor der kan ses på ledelsesformer.

3.3.2 Virksomhedens Menneskesyn

Det er vigtigt at være klar over, at menneskesynsteorien ikke handler om hvordan mennesket rent faktisk er, men om hvordan lederen opfatter sine medarbejdere. Ofte vil lederen have lidt fra både X-syn og Y-syn. I denne virksomhed har de tre ledere mere et Y-syn, da de har meget kontakt til deres medarbejdere, de holder f.eks. nogle forskellige arrangementer. Dog kan lederen Anders Olesen også ses som at et lidt X-syn, da han gerne vil have lavet salgsstatistikker over sine medarbejdere, og på denne måde for lidt kontrol over medarbejdernes indsats.

3.3.3 Lederstile

Hersey og Blanchards 4 lederstile skal sådan opfattes som 4 faser, alt efter hvor lang tid medarbejderen har været i virksomheden. Da de fleste i virksomheden har arbejdet der i mange år, vil de sandsynligvis ligge under den støttende ledelse.

3.3.4 Ledergitter

Ledelsen er 9.9 Holdlederen:

> *"Her udføres arbejdet af engagerede og motiverede medarbejdere. Spændende og udviklende arbejdsopgaver udføres for at nå nogle gensidigt accepterede mål. Her er trivslen høj, samtidig med at arbejdet opleves som værende fængende og motiverende".* [2]

Ledelsen går meget op i både sælge deres produkt og hvordan medarbejderen har det, og hvordan kunderne bliver behandlet, chefen bruger tid på at gå rundt i firmaet og motivere medarbejderne.

3.3.5 Autoritet

Autoritet fremkommer af 3 kilder, magt, kundskaber, viden og så Karisma. Anders Olesen bruger lidt magt idet, hvis han ikke er tilfreds med en medarbejder kan han være kontant, eller endda fyre medarbejderen, dog sker dette sjældent, men kan stadig ses som være en brug af magt. Det kan også tænkes at han også får lidt autoritet fra kundskab og viden, da han har været i branchen i mange år.

3.3.6 Adizes Lederroller

Anders Olesen hører under Integratorrollen, da han forsøger så godt som han kan at motiverer folk, og han inddrager forskellige folk i beslutningsprocesserne, hvilket viser at han leder gennem teamwork. Han går meget rundt i sin virksomhed og har et godt forhold til alle ansatte, hvilket viser at han Integrerer i et fællesskab. I forhold til at indgå kompromisser, har han ladet dem på kontorfunktionen have meget frihed, da han godt selv ved at det ikke er hans stærkeste side. Så på denne måde hører han meget godt under Integratorrollen.

Thomas Olesen høre også mere under Integratorrollen, han går rigtig meget op i fælleskabet i virksomheden, da han er formand i personaleforeningen og ofte arrangerer arrangementer.

Casper Olesen høre lidt mere under Producentrollen idet han oftere tager styring, og vil gerne hjælpe medarbejderne så opgaverne bliver løst rigtig og hurtigt. Som viser at han er flittig og mere resultatorienteret.

3.3.7 Virksomhedskultur type

Virksomheden kunne godt tyde på at være præget lidt af en Familie kulturtype, da der er lidt hierarki i virksomheden med Anders Olesen som den øverste og hans to sønner er begge ledere i hver sin afdeling. Samtidig er der et tæt forhold i virksomheden, da mange af dem har arbejdet der i lang tid, hvilket også gør at hvis de ikke klare sig helt så godt som Anders kunne tænke sig, skiller han sig sjældent af med dem.

3.4 Problemer, hypoteser, løsningsmuligheder

Oversigt over fundne problemer, hypoteser og løsningsmuligheder.

Problemer	Hypoteser	Løsningsmuligheder
Samarbejde med XL-Byg ikke fuldt integreret.	Da de har et forældet it system, som ikke er fuldt integreret til internettet og nethandel.	Et nyt fremtidssikret It-system.
UNIX system, som ikke lever op til kravene til informationer til styring af en virksomhed med en omsætning på omkring 69 mill. kr. pr. år.	Unix Systemet er forældet og kan ikke håndtere styringer når virksomheden bliver for stor.	Et nyt og forbedret it-system med god integration til internettet og nethandel.

Der anvendes ikke nuværende lokationsnumre	Da der ikke anvendes lokationsnumre, ved man reelt ikke hvor tingene befinder sig, eller om hvor den givne genstand hører til	En liste over genstande og lokationsnumre, kan hjælpe med at holde overblikket
Virksomheden har desuden udlejning af tæpperensere, gulvslibere og andre større værktøjer i alt ca. 30 enheder, som man skal holde styr på. Dette er hidtil kørt manuelt.	Da det gøres manuelt er der mulighed for menneskelige fejl, som tastefejl og glemsomhed.	Et nyt system kan holde styr på de forskellige udlejninger, samt give besked om hvornår det skal afleveres tilbage.

3.5 Vurdering af forskellige faktorer og opsamling i SWOT

Beskrivelsen af virksomhedens situation er sammenfattet i nedenstående SWOT matrice. SWOT(Strengths, Weaknesses, Opportunities, Threats) bruges som en fremgangsmåde til at opsamle og strukturere resultater fra andre analyser.

Intern	
Styke(Strengths)	Svagheder(weaknesses)
• 29 års erfaring • 1000 faste kunder • Virksomhed med en omsætning på omkring 69 mill. kr. pr. år. • Aktiekapital er på 5 mill. kr. • Trofaste medarbejdere	• Uvidende om varernes placering • Forældet It-system(UNIX) • Samarbejde med XL-Byg ikke fuldt integreret.
Ekstern	
Muligheder(Oppertunities)	Trusler(Threats)
• Handle via internettet • Størrer samarbejde • Flere kunder	• Voksende e-handel • Konkurrenter

4. Strategi Analyse

> *"Strategi er fastlæggelse af et mønster af handlinger, der sikrer, at virksomheden når sine overordnede mål gennem øget konkurrenceevne og merværdi."* [2]

Ud fra SWOT kan udledes følgende mulige strategier.

1. Maxi-maxi: Maximal udnyttelse af styrker og muligheder
2. **Mini-Maxi: Fokus på at minimere svagheder og maximere muligheder.**
3. Mini-Mini: Fokus på at minimere svagheder og trusler
4. Maxi-Mini: Fokus på at maksimere styrker og minimere trusler ...

Valg af strategi
Ud fra de mulige strategier vurderede gruppen at **Mini-Maxi** strategien, var den som ville give det bedste udbytte for virksomheden. Da vores opgave var at lave et nyt It-system, stemte punkterne i SWOT'ens svagheder og muligheder godt overens med det produkt som virksomheden ønskede.

Hvilke valg skal træffes omkring markeder og produkter?
Ifølge Ansoffs vækstmatrix har en virksomhed 4 mulige strategier for at opnå vækst:

1. Markedspenetrering - Det sker sjældent, at denne ikke bliver brugt, da den går ud på at øge markedsandel, hvilket alle virksomheder sigter efter ligemeget hvad.

2. Markedsnæstenudvikling - De får et nyt it-system som indeholder en net butik, og får derved potentielt udvidet deres marked, så de også når ud til nye segmenter.

3. Produktudvikling - Nyt produkt til samme marked. Det kan være en stor fordel for virksomheden, at være først på markedet med nye produkter inden for byggemarkedet.

4. Diversifikation - Opkøb af konkurrenter/leverandører, opstart af datterselskab

Ud fra de 4 ovenstående strategier indenfor intensivering, kan vi konkludere at Vestbjerg byggecenter A/S anvender sig af Markedspenetrering. De søger som de fleste andre virksomheder, at udvide deres kundekreds og derved øge deres markedsandel. Derudover ønsker de, at deres nye system skal tiltrække nye kunder, og forbedre deres konkurrenceevne.

4.1 IT Strategi

IT strategien skal understøtte forretningsstrategien. I forretningsvisionen beskrives hvordan applikationerne skal håndteres ud fra et forretningssynspunkt. Dernæst beskrives hvordan visionen opnås gennem beskrivelse af behovet for applikationer, information og teknologi. Endelig skal beskrives hvordan strategien skal udmøntes i organisationen, hvem der er ansvarlig, hvem der berøres osv.

4.1.1 Applikationer og information

På kort sigt: Fokus på salgs- og lagerstyring. Det skal være nemt for medarbejderne at bruge systemet, og det skal være effektivt i at holde styr på varerne og salgene.

På langt sigt: Der ønskes et system, hvor kan holde en statistik over kunder, varer, leverandører samt medarbejdere, og et system til udlejning af deres værktøj.

4.1.2 Teknologi

Selve systemet skal ligge på en server, der er tilkoblet et netværk, som firmaets computere kan tilgå. Derudover er det anbefalet, at firmaets computere ikke er alt for forældede, og er dette tilfældet, burde der anskaffes nyt udstyr.

4.1.3 Administration af IT-funktionerne

Vestbjerg kan have fordel i at ansætte en it afdeling(antal personer kan ansættes efter behov), til at holde styr på og vedligeholde deres nye it system, da de ville kunne bidrage med ekspert viden og erfaring inden for området. Firmaet kunne muligvis ansætte en IT-Supporter, som har en startløn på 24.837 kr. om måneden for 37 timer om ugen, eller ca 168 kr. i timen [**3**]. Da firmaet ikke har en specielt stor it-afdeling, kunne personen ansættes på deltid, eller blot ansættes efter brug.

4.1.4 Udvikling af systemet

Det kan være svært at vurdere, hvad systemet kommer til at koste firmaet at få udviklet. En datamatiker har i gennemsnittet en startløn på 32.504 kr. om måneden for 37 timer om ugen, eller cirka 220 kr. i timen [**3**]. Vores vurdering ud fra vores forholdsvis mindre erfaring, at udvikle et fuldt system i denne størrelse, ville tage en 4 måneders tid, med 4 ansatte datamatikere. Hvilket i sidste ende ville koste firmaet cirka 520.064 kr. i løn til datamatikerne.

4.2 It-handlingsplan

Handlingsplanen er lavet med ugangspunkt i [**1**]. IT-handlingsplanen giver et samlet overblik over hvor det bedst kan betale sig for virksomheden at foretage IT-investeringer. Udgangspunktet er, at der kun skal købes IT-værktøjer, der hvor det kan betale sig.

Vurdering af IT-behov: Det vurderes at, hvis firmaet fortsat skal være konkurrence dygtige, skal der investeres i et nyt IT system.
Hvordan kan IT-værktøjer afhjælpe de styringsmæssige udfordringer: Når de får et nyt of forbedret system, kan det hjælpe med at holde overblikket. Der bliver nemmere at holde styr på Ordre, Lager, hvilke medarbejder sælger meget, hvilke kunder køber meget og deres udlejning af værktøj.
Prioritering af IT-behov: Der laves et samlede system, hvori alle deres it-behov gerne skulle dækkes.
Vurdering af, hvor IT-værktøjer bør implementeres: Implementering af it-værktøjer kan ses på Parker/benson matricen figur 4.1.
Vurdering af potentialet i IT-projekterne i forhold til arbejdsprocessen: Ud fra vores vurdering er de projekter med højst potentiale også dem som har den længeste arbejdsprocess.
Vurdering af risiko i IT-projekterne: Risiko kan ses på Cost/Benefit delen i rapporten.
Prioritering af IT-projekterne : Prioriteringen af projekterne bliver gennemgået senere i rapporten.

4.3 C/B analyse

Til Cost/Benefit analysen har gruppen valgt at bruge Parker/Benson matricen, da den giver et godt overblik over, hvorvidt det system, som vi skal implementerer er de penge værd, som Vestbjerg Byggecenter A/S vælger at smide i det.

4.3.1 Parker/Benson matricen

	Business domain						Technology domain				Total
	ROI	SM	CA	MI	CR	OR	SA	DU	TU	IR	
Vægt	+5	+3	+4	+4	+1	-1	+2	-1	-2	-1	
Unix	2	2	2	1	2	0	0	0	0	0	30
Ordre styring	4	4	4	3	3	2	5	3	2	1	63
logistik/Lagerstyring	4	4	4	3	3	1	4	3	2	2	61
Produkt/kunde/medarbejder/leverandor database	1	2	3	1	0	2	3	0	2	1	26
Statistik	1	4	3	4	2	0	4	0	0	1	54
Udlejning	1	1	1	0	0	0	1	1	0	1	12
E-handel	2	4	4	2	3	5	3	3	3	3	38

Vurderingsfaktorer:

ROI = Return of investment
SM = Strategic match
CA = Competitive advantage
MI = Management information
CR = Competitive response
OR = Organizational or project risk
SA = Strategic IS architecture
DU = Definitional uncertainty
TU = Technical uncertainty
IR = IS infrastructure risk

Figure 4.1: Parker/Benson Matricen

Diverse dele af systemet blev delt op, for at give et overblik over, hvad der bedst vil kunne betale sig at lægge vægt på. Som det ses på figur 4.1 er ordre styring og lagersttyring i top, hvor statistik ligger meget tæt på. Dette gør det også nemmere, at vælge hvilken use case der skal fokuseres på senere, hvor gruppen valgte Ordre styring og lager styring.

4.4 Implementering

Implementeringen foregår i iterationer, hvor vi tager fat i én usecase ad gangen, og implementerer den. Her startes med den mest komplekse use case, og derefter den næst mest komplekse o.s.v.

4.5 Konklusion på forundersøgelse

Vestbjerg byggecenter har et forældet it-system kaldet UNIX, der ikke lever op til de krav der er omkring information til styring af en virksomhed af deres størrelse. De ønsker at få et nyt og bedre system, og på baggrund af dette har gruppen lavet en it-forundersøgelse. Gruppen kan gennem it-forundersøgelsen konkludere, at det er muligt for at programmere det nye system som opfylder disse krav omkring informationsstyring, og at der derudover også kan betale sig for Vestbjerg Byggecenter at betale os for at lave det nye system.

5. Inseption

Inseption er den første del af system design, som indeholder Medarbejder mål-tabel, workfow, domain model og fulle dressed beskrivelse.

5.1 Medarbejder mål-tabel

Denne tabel beskriver en medarbejders opgave, hvad målet er med opgaven og til sidst beskriver den de forskellige steps i opgaven. Dette er den som man starter ud med, når man begynder på system udvikling, og den giver et overblik over, hvad et system muligvis skal indeholde.

Medarbejder	Opgave	Mål	Steps i opgave
Medarbejder	Oprette nyt produkt	Oprettet et nyt produkt i systemet	1. Tjekke om produktet er i systemet 2. Registrere nyt produkt
Sælger	Ordre af produkt	Ordre af produkt til kunde	1. Tilføjelse af Ordre 2. Opdatering af produkt i lagerbeholdningen
Sælger	Reservation af produkt	Reservation af produkt til kunde	1. Informationer indtastes 2. Kunde oprettes i system 3. Kunde bliver tildelt produkt
Medarbejder	Tjekke lagerbeholdning	Tjek af lagerbeholdning	1. Find det ønskede produkt 2. Tjekke lagerbeholdningen 3. De hentede oplysninger printes
Medarbejder	Bestil varer	Bestil nye varer når beholdning er lav	1. Finde det ønskede produkt 2. Tjekke lagerbeholdningen 3. Bestil nyt hvis der mangler
Sælger	Oprette ny Kunde	Oprettet en ny kunde	1. Registrer kunde i systemet
Lagerarbejder	Pakke Produkt	Få pakket produktet	1. Find ordren 2. Pak ordren

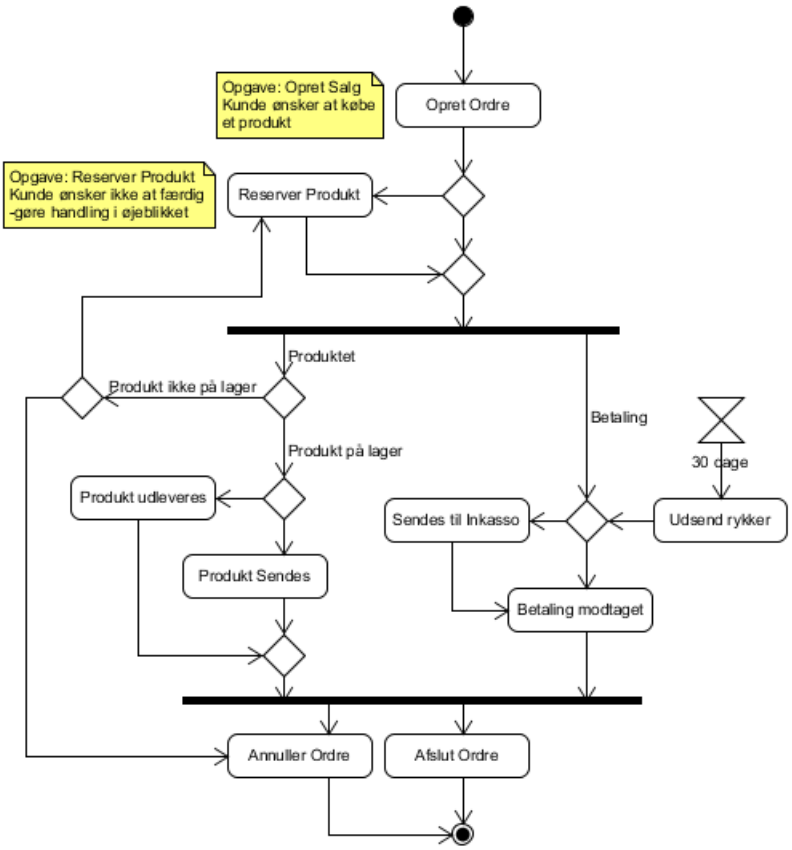
Lagerarbejder	Levere Produkt	Sende produktet til levering	1. Find ordren 2. Tjek ordren er pakket 3. Gør ordren klat til levering 4. Lever ordren
Chef	Print statistikker	Udprintning af statistikker mht. kunder, leverandører, varer og medarbejdere	1. Informationer hentes i systemet 2. De hentede informationer printes

5.2 Workflow

Workflowet på figur 5.1 er for use casen opret ordre, som viser alle steps som finder sted.

Hver firkant viser et step, og pilene viser vejen til det næste step. Den sorte firkant er når systemet kan gå flere veje, i dette tilfælde er der en vej for produktet, og en vej til betalingen. Den hvide firkant er til at vise når et step, kan fører til flere andre steps, eksempelvis med produktet der enten er på eller ikke på lageret, som vil give et nyt step. Til Sidst er der steps hvor ordren enten kan annulleres eller afsluttes, annuleringen kan kunden også foretage hvis produktet ikke er på lager.

Figure 5.1: Workflow



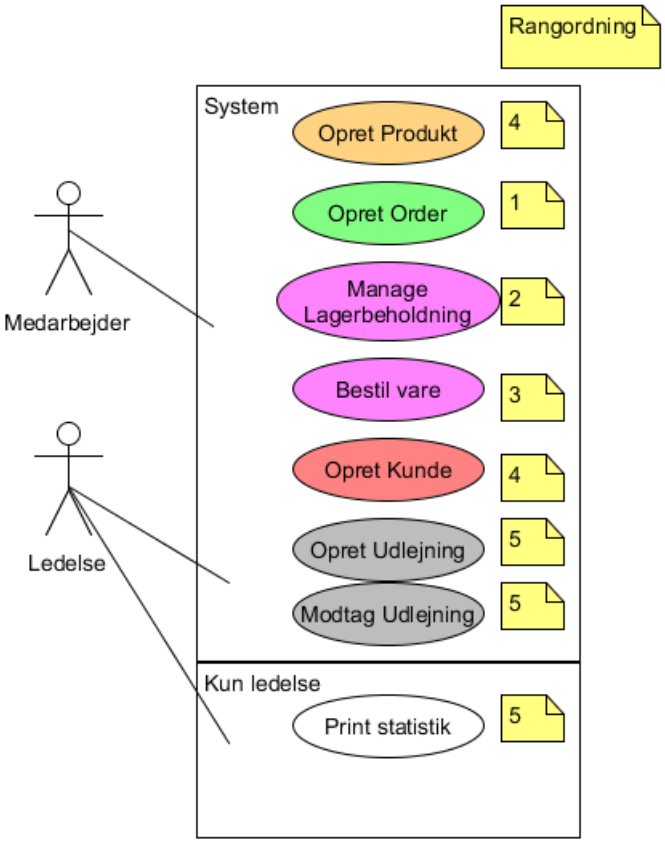
5.3 UseCase

På figur 5.2 ses listen over use casene, og et nummer som repræsentere deres rang. Farverne på use casene er til for at gruppere dem, så farveren passer til de use case som hører mere eller mindre sammen.

Rangordningen er placeret ud fra Parker/Benson matricen, som er en samlet vurdering af forskellige faktorer, og en vurdering ud fra hvilken use case der er den mest komplekse. Det blev Opret order som er den use case, som gruppen vurderede var den mest komplekse.

På figuren 5.2 ses en medarbejder og en fra ledelsen. Det hele er et samlet system, dog med en enkelt use case som kun ledelsen kan se. Stregerne viser hvordan de to aktører, kan bruge systemet, og hvad de har adgang til. Medarbejderen har eksempelvis ikke adgang til statistik, som er noget ledelsen kun skal se og gøre nytte af.

Figure 5.2: Use Case



5.4 Domain Model

Domæne Modellen som ses på figur 5.3 har 7 forskellige klasser som hver har en række attributter og associeringer. Det kan f.eks. ses at en Person har nogle attributter som beskriver en Person, som navn, address og Phone, ligeledes har Product nogle attributter som beskriver den, som Name, price og weight.

Det kan ses at både customer og coworker arver fra superklassen Person, hvilket gør de har de samme attributter som Person. Customer og Coworker har begge en associering til klassen Order, Order kan kun have 1 Customer og 1 Coworker, mens de begge kan have 0 til mange Order. delOrder klassen har en aggregering til klassen Order, hvilket betyder at de har en tæt forbindelse til hinanden. Her kan en Order have 0 til mange delOrders, mens en Order kun har 1 Order. En delorder kan have 1 Product hvor et Product kan have 0 til mange delOrder. Til sidst har vi at Inventory kan have 0 til mange Prooduct, og Product kan

have 1 Inventory.

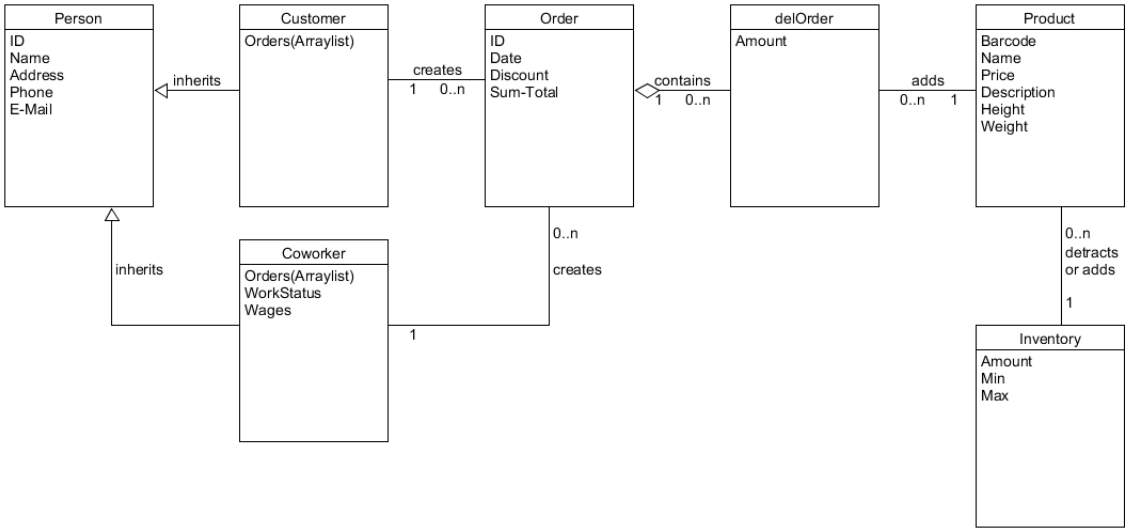


Figure 5.3: Domain Model for vestbjerg byggecenter

5.5 Fully Dressed

En fully dressed beskrivelse af use case Opret Ordre(Create Order). I denne tabel vil der være en detaljerede gennemgang af opret ordre, hvor hver enkelt step vil blive beskrevet. Alternative flows kommer sidst i tabel, hvor den viser hvilke andre veje kunne forekomme i systemet.

Use Case navn	Opret Ordre	
Aktører	Medarbejder	
Pre-Betingelse	Der er oprettet produkt(er), kunde(r) og medarbejder(er)	
Post-Betingelse	En ansat har færdig gjort ordren til en kunde	
Flow of events	Aktør handling	System
	1. En kunde vil købe nogle produkter	
	2. Den ansatte starter en ordre	3. Systemet registrerer ordren

	4. Den ansatte taster informationer de nødvendige information som, kunde id, ansattes id, date, rabat, produkt stregkode, antal	5. Systemet registrerer de givende information og opretter en ordre
	6. Den ansatte tjekker ordren og afslutter den	7. Systemet registrerer til sidst ordren og associerer ordren med kunden og medarbejderen
Alternative flows		
	3.a. Produktet kan ikke findes	Systemet sender fejlmeddelelse
	3.b. Produktet er ikke på lageret	Systemet sender fejlmeddelelse
	6.a. Betalingen er ikke modtaget	Systemet printer/sender en rykker

6. Elabration

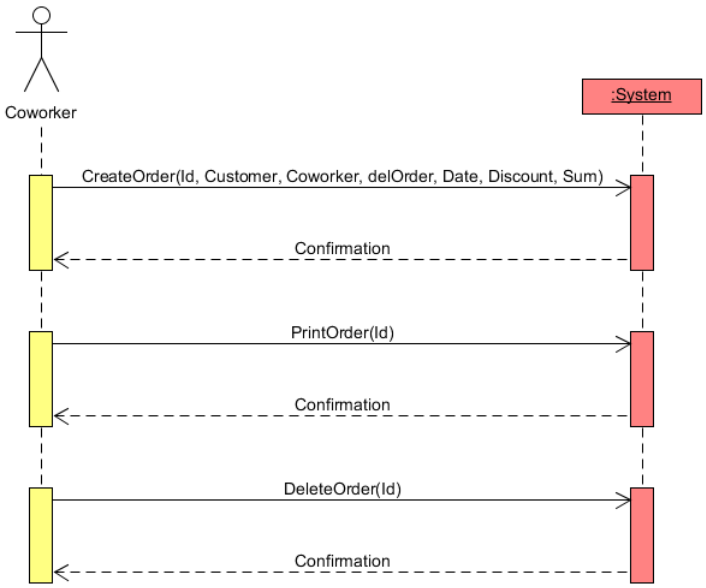
Dette er den sidste del af system udviklingen, og den indeholder SSD, Kommunikations diagrammer, design klasse diagrammet og programmeringen.

6.1 System Sequence Diagram(SSD)

Et SSD(system sekvens diagram) viser en række forskellige handlinger som de vigtigste use cases gør. Den viser Aktøren, systemet, og forskellige system hændelser som aktøren kan fortage sig, og hvad systemet eventuelt svarer tilbage til aktøren.

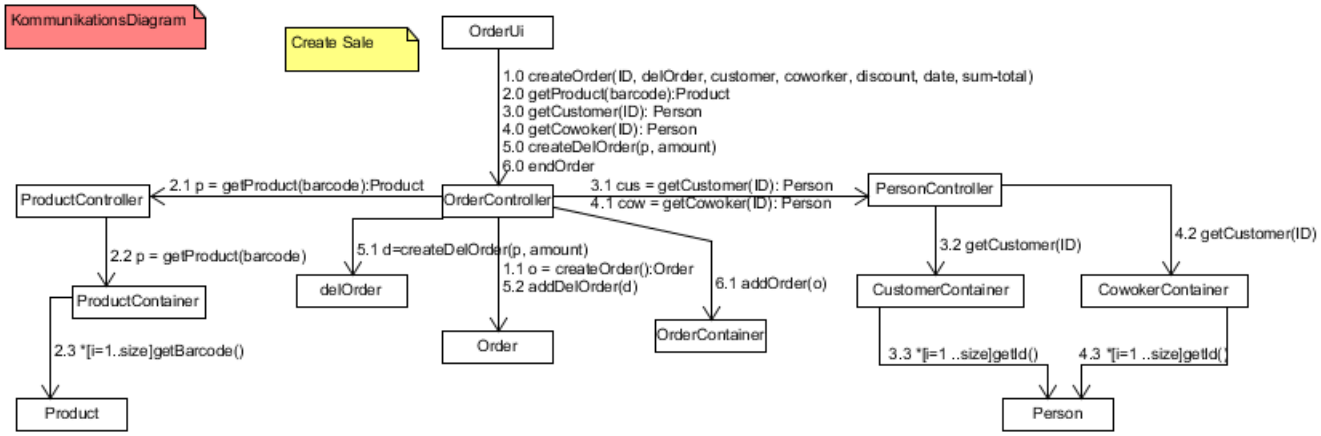
SSD'en som ses på figur 6.1 er til håndteringen af en Order, som består af en CreateOrder, som tager en række informationer der skal bruges, som et id, customer coworker osv. og får så en meddelelse tilbage fra systemet om at ordren er oprettet. PrintOrder hvor aktøren skriver id ind på den ordre der ønskes, systemet printer informationen om Ordren ud. DeleteOrder der også tager et ID, og systemet giver en besked om at Ordrern er slettet.

Figure 6.1: SSD over CreateOrder



6.2 Kommunikations Diagram

Figure 6.2: Create Order Kommunikations Diagram



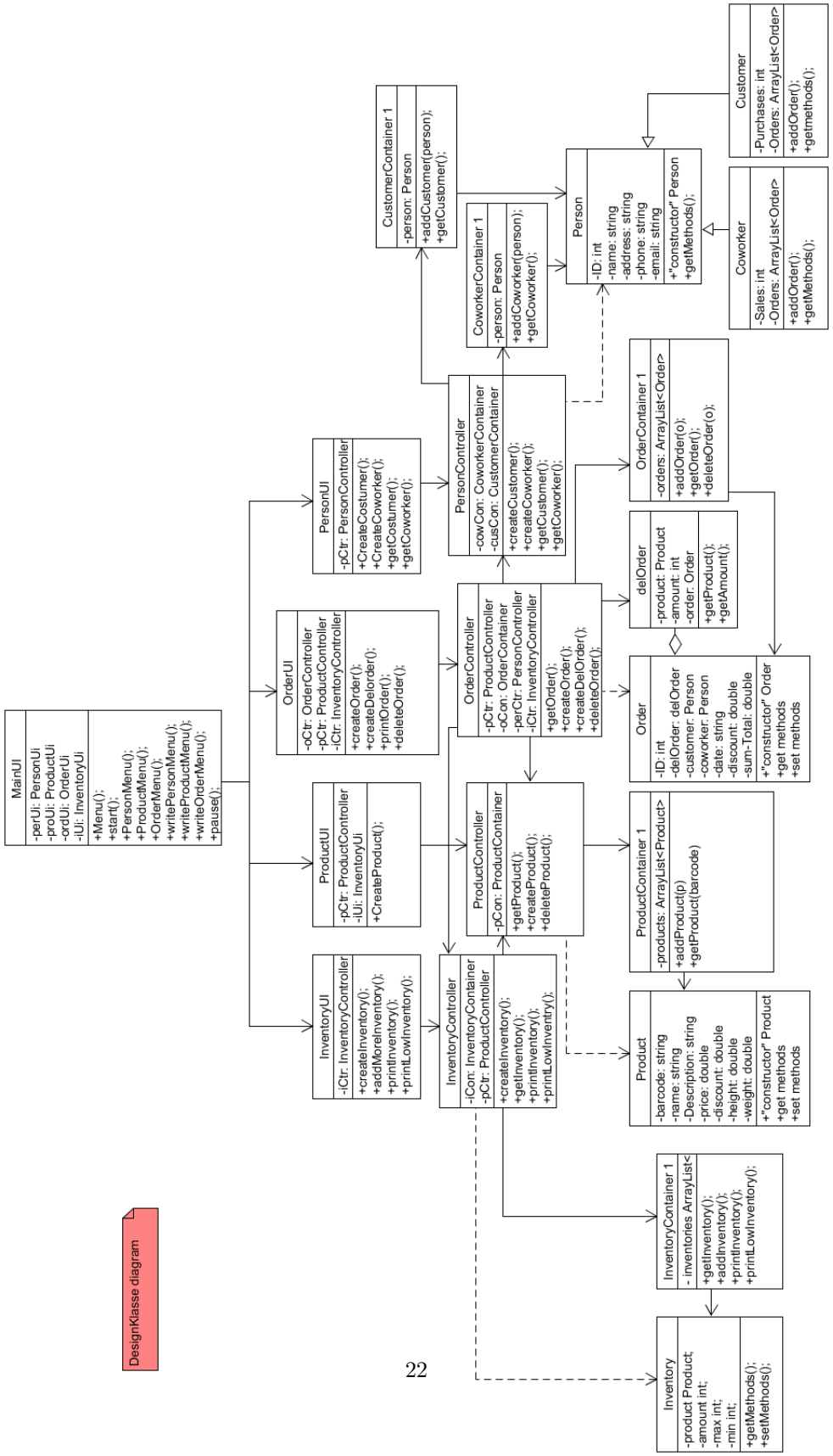
Et kommunikations diagram er et dynamisk diagram, idet det viser hvordan klasserne kommunikere i mellem hinanden. Det bliver lavet ud fra et bestemt use case, og derfra skal det vise hvordan kommunikationen sker, for at use casen kan blive udført. På diagrammet kan man se navnet på klassernes metoder, og hvilke parameter de har med.

På figur 6.2 ses kommunikations diagrammet for createOrder use casen, heri vises hvordan en ordre bliver oprettet, og hvordan klasserne "snakker" sammen for at oprette en ordre. Dette er en af de mere komplekse use cases, hvilket kan ses på diagrammets størrelse, og hvor meget det skal rundt i systemet.

6.3 Design Klasse Diagram

Design klassediagrammet bruges, så man får en helhedsfornemmelse over, hvad systemet skal indebære. De forskellige klasser fra de forskellige lag bliver sat op med hinanden, og forbundet, så man kan se, hvad de forskellige metoder gør, trin-for-trin. Derfor gør design klassediagrammet det nemmere at kode systemet, da man ved hjælp af dette kan se sammenhængen mellem de forskellige klasser og de metoder, som skal implementeres.

Figure 6.3: DesignKlasseDiagram



6.4 Arkitektur

Systemet bruger et 3 lags arkitektur system, der består af UiLayer, ControlLayer og ModelLayer. Dette gør systemet mere overskueligt når det skal programmeres, og giver en god sammenhæng i mellem klasserne.

UiLayer: I dette lag er der klasser som OrderUi, der tager information ind fra brugeren og sender det videre til en klasse i ControllerLayer, eksempelvis vil OrderUi sende informationen videre til OrderController.

ControllerLayer: I dette lag modtages informationen fra UI og kalder de forskellige metoder. Der er f.eks. en klasse i ControllerLayer der hedder OrderController, denne klasse kalder de metoder der har med Order at gøre, som at createOrder, og skal den bruge andre metoder såsom getCustomer, kalder den PersonControlleren, som så kalder metoden getCustomer.

ModelLayer: I dette lag er alle grund klasserne, såsom Product, Order, Inventory osv. med deres tilhørende containere.

7. Java koden

Kode eksempler fra gruppens system vil blive vist og beskrevet i dette afsnit, og med en forklaring på hvorfor det er brugt.

7.0.1 Hvad programmet kan

Der kan oprettes produkter, hvor det også bestemmes hvor mange af den type der produkt skal være på lageret, hvor mange der max må være, og hvad minimum er før der skal bestilles nye, og hvor dette product placeres på lageret.
Der kan oprettes personer, ad typen customer og coworker.
Der kan oprettes Ordre med delOrdre(med produkt og antal), hvor der kan være flere delOrdre på en ordre, så flere produkter kan være på en ordre, en ordre kan også slettes.

Der er et simpelt lager hvor i produkter bliver lagt, og herfra bliver produkterne også trukket fra når de ligges på en ordre, så derfor kan der heller ikke ligges på ordre, når der ikke er nok på lageret.
Der kan bestilles nye produkter til lageret.
Samtidig er der lavet nogle if statements der fanger hvis programmet prøver at hente noget der ikke eksitere, som f.eks. man indtrykker et forkert ID til en costumer.
Liste over ting fra koden/systemet, og hvorfor det er brugt

- Singleton: I systemet er der brugt singleton til klassernes containere. Det sikre at, der kun bliver oprettet en instans af klassen. Hvis der ikke er singleton, kan en klasse oprette en ny instans af klassen, og derved vil der fås en nullpointer exception, da der ikke vil være puttet noget i containeren.

- ArrayList: I hver container er der lagt en ArrayList i, fordi det vil være nemmere at gemme og hente data fra listen end f.eks. fra en LinkedList. Der er også arraylist i andre klasser, så som Ordre, da denne skal kunne have delordre i sig.

- While loops: Er brugt til en search funktion. Den er blevet brugt til at gå listen med elementer igennem, og den stopper først når den søgte element er fundet.

- For Each loop: Dette er en sweep, som er blevet brugt til at f.eks. printe alle delOrders ud på en order, fordi den går igennem alle elementer på en ArrayList.

- Superklasse Person: Der er brugt en super klasse kaldet person, hvorfra klasserne Coworker og Costumer arver. De to klasser har attributer som er fælles for dem begge, og nogle få som er unikke. Herved spares noget kodning, og der kan altid laves flere klasser som eksempelvis arver fra medarbejder, hvis der skal være forskel på medarbejderne f.eks. lager, sælger og ledelse.
- MainUI: I UILayeret er der lavet en MainUI klasse, det er denne som der skal startes, og fra denne kan alle de andre menuer tilgås. Dette er til for at have en overskuelig tilgang til menuerne.
- Scanner: Når der skal læses input fra brugeren, bliver java værktøjet Scanner hentet. Denne funktion læser hvad brugeren taster ind, og dette kan så gives til en variabel.
- Iterator: Denne er blevet brugt til at slette elementer fra f.eks. en container.
- Abstract: Dette er blevet brugt til person klassen, hvilket gør at den ikke kan blive instantieres. Derimod er man nødt til at oprette en customer eller coworker, som arver fra person.

7.0.2 Eksempel på Switch menu

Der blev brugt switch statements til at lave menuer i systemet, som kan ses i figur 7.1. Switchens cases tager en int som værdi, der bliver returneret fra writeMainMenu(), og udføre indholdet i den givne case.

Switch har en break i hver case(undtagen default), når den læses hopper den ud af switchen, og skal derfor ikke nødvendigvis igennem hver eneste case.

Switch er hurtig og overskuelig, og passer godt til at lave simple menuer, som dem der skal bruges i systemet.

```
public void Menu()
{
    boolean exit = false;
    while(!exit)
    {
        int choice = writeMainMenu();
        switch(choice){
            case 1: PersonMenu();
                break;
            case 2: ProductMenu();
                break;
            case 3: OrderMenu();
                break;
            default: exit = true;
        }
    }
}
```

Figure 7.1: Switch Menu

7.0.3 Create Order fra controller

```
public void createOrder(int id, int idCu, int idCo, String date, double discount)
{
    Order o = new Order(id, peCtr.getCustomer(idCu), peCtr.getCoworker(idCo), date, discount);
    oCon.addOrder(o);
    peCtr.getCoworker(idCo).addOrder(o);
    peCtr.getCustomer(idCu).addOrder(o);
}
```

Figure 7.2: Create Order

Firgur 7.2 viser createOrdre metoden, som tager 5 parametre. Der bliver oprettet et nyt objekt af Order kaldet o, hvor værdierne fra parametrene bliver givet til objektet. Herefter kaldes addOrdre metoden fra klassen OderContainer(oCon), ordren bliver gemt i en arrayliste. Til sidst kaldes to metoder, den ene fra coworker og den anden fra customer, og ordren bliver derved gemt i en arraylist på den givne coworker og customer.

7.0.4 Create DelOrder fra controller

```
public void createDelOrder(int amount, String barcode, int id)
{
    DelOrder del = new DelOrder(amount, pCtr.getProduct(barcode), oCon.getOrder(id));
    oCon.getOrder(id).addDelOrder(del);
}
```

Figure 7.3: Create DelOrder

På figur 7.3 ses createDelOrdre metoden, som tager 3 parametre(int amount, String barcode og int id). Der bliver oprettet et nyt objekt af DelOrder kaldet del, hvor værdierne fra parameterne bliver tildelt objektet, for at få fat i et product med bacode, kaldes pCtr.getProduct() som et en metode i ProductController, og ligeledes kaldes oCon.getOrder(). Bagefter kaldes oCon.getOrder(id).addDelOrder(del) fra ordre containeren som henter det bestemte order, og det er denne order hvori delOrdren(del) bliver lagt.

7.0.5 Create Order fra UI

```
public void CreateOrder()
{
    System.out.println("Enter Id ");
    int id = keyboard.nextInt();

    System.out.println("Enter Customer ID ");
    int idCu = keyboard.nextInt();
    if(peCtr.getCustomer(idCu) == null){System.out.println("The customer id does not exist");}
    else{
        System.out.println("Enter Coworker ID ");
        int idCo = keyboard.nextInt();
        keyboard.nextLine();
        if(peCtr.getCoworker(idCo) == null){System.out.println("The Coworker id does not exist");}
        else{
            System.out.println("Enter date ");
            String date = keyboard.nextLine();

            System.out.println("Enter discount ");
            double discount = keyboard.nextDouble();

            boolean dis = false;
            while(!dis){
                if(discount > 30 && !peCtr.getCoworker(idCo).getWorkStatus().equals("Chef"))
                {
                    System.out.println("you cannot make a discount bigger than 30%");
                    System.out.println("Enter discount of 30 or lower");
                    discount = keyboard.nextDouble();
                }
                else{dis = true;}
            }
            oCtr.createOrder(id, idCu, idCo, date, discount);
            System.out.println("you Order has been created with the id " + id);
boolean state = false;
while(!state){
    oCtr.getOrder(id);
    keyboard.nextLine();
    System.out.println("Enter Product barcode ");
    String barcode = keyboard.nextLine();
    if(pCtr.getProduct(barcode) == null){System.out.println("The Product barcode does not exist"); state = true;}
    else{
        System.out.println("Enter amount ");
        int amount = keyboard.nextInt();
        if(iCtr.getInventory(barcode).getAmount() >= amount){
            oCtr.createDelOrder(amount, barcode, id);
            iCtr.getInventory(barcode).detractAmount(amount);

            System.out.println(" Current amount of products left in the inventory =  " +
            iCtr.getInventory(barcode).getAmount() );
            System.out.println(" Amount of products available for sale before an order for new products will be made = " +
            (iCtr.getInventory(barcode).getAmount() - iCtr.getInventory(barcode).getMin()));

            System.out.println("Press 1 to add another product, 2 to stop");
            int choice = keyboard.nextInt();
            if(choice == 2){state = true;}
        }
        else{System.out.println("You dont have enough amount of products, you only have: " +
            iCtr.getInventory(barcode).getAmount()); state = true;}
```

Figure 7.4: CreateOrderUI

På figur 7.4, ses hvordan en ordre bliver lavet i UI klassen. Først indtastes et ID, derefter indtastes en kunde(customer) ID, som er efterfulgt af en if statement, hvor den læser om ID'et findes. Derefter en medarbejder(coworker) ID, som også tjekker om ID'et findes, og derefter en dato. Bagefter skal der indtastes en rabat(discount), med denne følger et while loop, som tjekker om medarbejder er Chef, hvis ikke den er lig med sandt kan der gives en større rabat, hvis ikke kan den højst være 25 procent. Efter dette bliver selve ordre objektet lavet, så nu er ordren lavet.

Herefter begynder et while loop, hvori der kan tilføjes produkter til ordren, herefter en if statement som tjekker om produktet findes. Den er i et while loop, så der kan tilføjes produkter så længe man lyster. Først hentes Ordre ID'et, som blev lavet ovenfor. Herefter bliver produkt indtastet(hentes fra en anden container), sammen med det ønskede antal. Dette efterføles af en if-statement, som tjekker om der er nok af produktet på lageret(inventory), hvis sandt trækker den det givet antal fra antallet på lageret. Derefter en if-statement som tjekker et indtastet tal, hvis indtastet 2 hopper den ud af loopet, og der ordren er færdig. Til sidst er der en else-statement, som bliver udført hvis der ikke er nok af produktet på lageret.

7.0.6 getOrder

```
public Order getOrder(int id)
{
    Order order = null;
    int index = 0;
    boolean found = false;
    while(index < orders.size() && !found) {
        order = orders.get(index);
        if(order.getId()== id) {
            found = true;
        }
        else {
            index++;
        }
    }
    if (found) {
        return orders.get(index);
    }
    else {
        return null;
    }
}
```

Figure 7.5: getOrder

På firgur 7.5 ses koden for getOrder, denne vises da "get" bruges flere stedder i programmet. I dette tilfælde tager den en int id parameter, dette kunne også

være String, men en Order skal søges efter dens ID da dette er unikt. I selve metoden sætter vi order til null, en int indes til 0, og en boolean found til false. Herefter laves en while, der køre så længe at index er mindre en vores arraylistes størrelse(orders.size()) og at vores boolean ikke er false. Herefter sættes order lig den order som er på vores index nummer, som her er 0, hvilket betydder den tager det aller første order. Så er der en if, som tjekker om denne order's id er lig det id som er tastet ind i parameteren, er den det sættes boolean til true, og vi kommer ud af while loopet, er den ikke det, kommer den ned i else hvor der er index++, som gøre at 1 lægges til index. Tilsidst er der en if(found) som gør, at hvis boolean found er true, returneres Order med det øsnket id, ellers returneres null.

8. Konklusion

Gruppen har ved hjælp af de forskellige fremgangsmetoder fået lavet et stykke arbejde, for vestbjerg. Der blev i starten fundet frem til at vestbjerg byggecenter havde et forældet system, som skulle fornys, da det ikke levede op til de forskellige krav. Gruppen havde ud fra parker/benson matricen fundet ud af, at det ville være bedst at fokusere på Ordre styring samt lagerstyring. Herefter udarbejdede gruppen diverse diagrammer såsom use case, SSD, kommunikations diagrammer og design klassediagram. Dette er for at få et overblik over hvad programmet skulle indeholde af metoder og attributter, hvor gruppen derefter kunne begynde at kode selve programmet. Programmet indeholder derved en ordrestyring, hvor der kan oprette ordre, samt printe den ud og slette den igen. Den indeholder også en lagerstyring, når du opretter et nyt product bliver der også oprettet et lagerdertil, som har et antal a producter, samt min og max antal, og en placering på produktet.

8.1 Gruppe Evaluering

Gruppearbejdet er gået godt. Gruppe arbejdet foregik oppe på universitetet, hvor gruppen mødte op hver dag, medmindre andet var blevet aftalt. Dette gjorde det nemmere at snakke sammen om tingene, og hjælpe hinanden med de forskellige opgaver, og give gruppens medlemmer mere erfaring inden for gruppe arbejde i sådan et type projekt. Gruppen havde fået tildelt en fælles skærm, som blev gjort meget nytte af, og gjorde det nemmere når der skulle laves noget sammen.

8.2 Tidsplan

Vedlagt i bilag.
I starten lavede gruppen en tidsplan, som blev lavet efter de forskellige iterationer. På højre side står den oprindelige tidsplan som blev lagt, og til venstre står datoerne, vejlederne og en opdatering når gruppen var nået længere. Tidsplanen blev fulgt nogenlunde, men det gik lidt hurtigere end planlagt, og tingene blev hurtigere færdig end forventet.

8.3 Værktøjer

Gruppen valgte at bruge **sharelatex.com** til at skrive rapporten i, da medlemmerne kan skrive på samme tid, og redigere i hinandens tekster, og LaTeX er et godt stykke software til at skrive rapporter i. **Trello** blev brugt til at lave en tabel over hvad der skulle laves, så der kunne holdes styr på hvad der var færdigt, hvad der ikke var begyndt på endnu, og hvad der var i gang med.

8.4 Gruppe Kontrakt

Vedlagt i bilag.
Gruppekontrakten blev udarbejdet den første dag, og alle i gruppen var tilfreds med kontrakten. Der har dog ikke været problemer eller opstået konflikter i gruppen, så kontrakten har ikke rigtig blevet hevet frem på noget tidspunkt.

8. Bibliography

[1] Basit. It-handlingsplaner sikrer i rigtige investeringer. http://www.basit.dk/services/it-handlingsplan/.

[2] Perter Storm-Henningsen Hans Jørgen Skriver, Erik Staunstrup. *Organisation.* 5 edition, 2016.

[3] Prosa. Startløninger. https://www.prosa.dk/raadgivning/loenstatistik/startloenninger/.

9. System-Brugervejledning

Når programmet startes ses 3 menuer, 1. Person Menu, 2. Product and Inventory menu og 3. Order Menu. For at navigere i menuerne skal du indtaste et ønsket tal, f.eks. 1 hvis du vil ind i Person menu. Vil du tilbage skal du indtaste et tal som er højere end det er vises på nogle af menuerne som f.eks. i Person menu er der op til 4 menuer, her skal du trykke 5 eller derover, for at vende tilbage til Main Menu.

Trykker du på 1. kommer du ind i Person menuen, hvor du kan opretter Customers og og Coworkers, hvor du bagefter kan hente de informationer om disse personer, som du har skrevet ind.

Trykker du på 2. kommer du ind på product og inventory menu. Hvor der kan oprettes et Product, add more to inventory to a product, Print Inventory og Print low inventory. Den sidstnævnte fortæller, om der er for få produkter på lager, og om der derfor skal bestilles flere.

Trykker du på 3, kommer du ind i Order menuen, hvor du kan lave en Order. Dette gøres ved, at du skriver informationer ind om Coworker, Customer og Product, herefter er en Order lavet. Skal der pludselig flere produkter på Orderen, kan det også lade sig gøre

10. Bilag

Uge 47	
23/11 - 2016	Gruppekontrakt, Tidsplan
24/11 - 2016	It-forundersøgelse
25/11 - 2016	It-forundersøgelse
Uge 48	
28/11 - 2016 (**Ikl**) **Begyndte på:** Inseption	It-forundersøgelse Afsluttes
29/11 - 2016 **Begyndte på:** Domain Model	Inseption 1. MedArbejder - måltabel 2. Workflow 3. Mock Ups? 4. Use case diagram
30/11 - 2016 (**Ikl**) **Begyndte på:** Elabration for UseCase1 Opret Ordre	Inseption 1. Brief use case beskrivelse? 2. Ikke funktionelle krav? 3. Domain Model 4. Rangordning af use case 5. Fully dressed Inseption = færdig.
1/12 - 2016 (**guan**) **Begyndt på:** Kode Use case 1 næsten færdig	Elabration for Use Case 1. 1. SSD 2. Arkitektur 3. Kommunikations diagram 4. Design klasse diagram
2/12 - 2016 Use case 1 nogenlunde færdig	Elabration for Use Case 1

Figure 10.1: Tidsplan

Uge 49	
5/12 - 2016 (**guan**) **Begyndte på:** UseCase2 - lagerstyring	Elabration for use Case 1 = færdig
6/12 - 2016	Elabration for use Case 1 5. Kode 6. Test 7. TUI-Prototype
7/12 - 2016 (**hemh**) Arbejde hjemmefra	
8/12 - 2016 (**alcl**) Arbejde hjemmefra	Elabration Use case 1 færdig
9/12 - 2016 (**hemh**) Finpudser rapport og program	Construction - De sidste use cases
Uge 50	
12/12 - 2016 (**alcl**) Arbejder hjemmefra	
13/12 - 2016 (**hemh**)	Construction = færdig
14/12 - 2016	Finpudse rapport
	Før - 15:00 Afleveres Rapporten!!!

Figure 10.2: Tidsplan

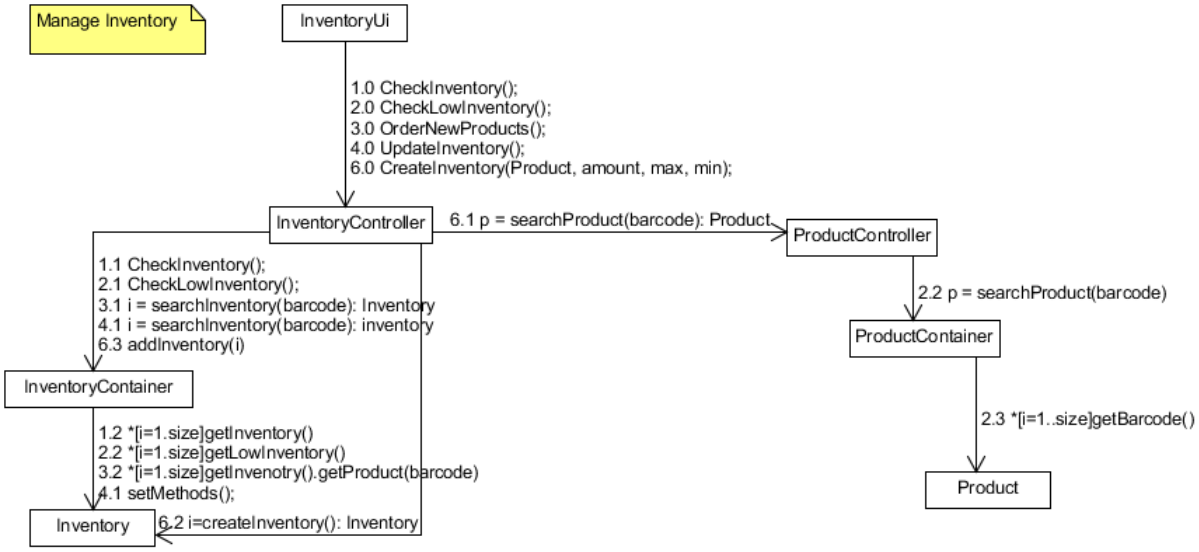


Figure 10.3: Inventory KommunikationDiagram

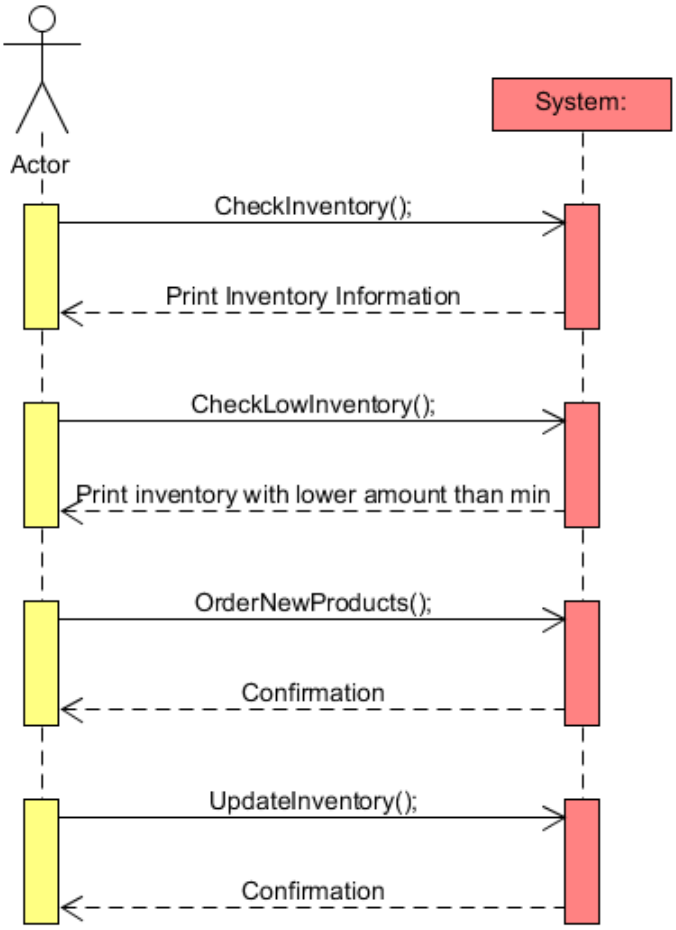


Figure 10.4: Inventory KommunikationDiagram

10.1 Gruppekontrakt for Gruppe 8

Mødetider og pauser

- Gruppen møder ind 08:30.
- Hvis man bliver forsinket eller ikke kan møde op, skal man kontakte et af de andre medlemmer i gruppen.
  - Frokostpause 11:30.
  - Små pauser styre man selv, og kan tages når man har brug for det.
- Møder man ikke som aftalt, uden at kontakte et gruppemedlem, skal man give kaffe og kage(gerne romkugler, og karat kaffe)?

Forventninger til projektet og gruppen

- Vi hjælper hinanden i gruppen.
- Der er ingen dumme spørgsmål, har man tvivlsspørgsmål så spørg.
- Arbejde målrettet med god/høj arbejdsmoral, og der arbejdes seriøst i gruppen.
- Sociale medier og mobil må kun benyttes i de aftalte pauser.
- Vi vil bruge vores vejleder til tvivlsspørgsmål og til hjælp i processen.

Konflikter

- Hvis der opstår konflikter, prøver vi at løse dem sammen i gruppen.
- Hvis en konflikt ikke bliver løst, tages der kontakt til en vejleder.

Navne, mailadresser og telefonnumre på alle medlemmer
Thais Gilkrog, tgilkrog@gmail.com. 51 93 33 45.
Insight: 32 - Supporterende Koordinator(Klassisk)

Marc René Jensen, marc.r.jensen@gmail.com 28 73 52 13
Insight: 31 - Koordinerende Supporter(Klassik)

Paw Gilkrog. pgilkrog@gmail.com. 21 53 55 74.
Insight: 34 - Koordinerende Observatør(Klassisk)

Jacob Elgaard Winther Nielsen, jacob.e.w@hotmail.com 53 61 88 65
Insight: 34 - Koordinerende Observatør(Klassisk)