



Erhvervsakademi og
Professionshøjskole



Overbliks-system til Fyens Børste- og Kostefabrik

UCL - Erhvervsakademi og Professionshøjskole
Datamatiker
DMOE22
Gruppe 9

Andreas Pedersen

anpe48593@edu.ucl.dk

Christian Hennie

chhe48827@edu.ucl.dk

Ihab Dabaan

ihda50386@edu.ucl.dk

Malthe Ingerslev

main48945@edu.ucl.dk

Oguz Yorulmaz

ogyo50908@edu.ucl.dk

Rasmus Enevoldsen

rnen48952@edu.ucl.dk

Rasmus Kallehauge

rwka48711@edu.ucl.dk

16. december 2022

Indhold

1	Indledning	1
2	Systembeskrivelse	1
3	Information om produkter, ordrer og lokationer	2
4	Design	11
5	Implementering	14
6	Business case	17
7	Proces	21
8	Konklusion	23

1 Indledning

Hvad har vi egentlig liggende på lager? Hvordan holder vi overblik over et varelager, ordreflow og samtidig holder det opdateret?

Disse spørgsmål udarbejder den problemstilling, som vi vil besvare i denne rapport samt i vores prototype. Baseret på information fra vores product owner i virksomheden, samt observationer af de relevante arbejdsgange, har vi dannet et overblik over en løsning, som stemmer overens med de ønsker der er blevet stillet af virksomheden.

Igennem denne rapport vil man få et indblik i de designvalg, implementeringer m.m. som vi har valgt og den proces vi har været igennem - men også har fravalgt undervejs i forløbet.

Virksomheden hedder Fyens Børste- og Kostefabrik (FBK). De producerer og sælger industriprodukter til det professionelle marked, med fokus på fødevare-, hospital- og kemi-industrien. De sælger til alle verdens kontinenter og har et varelager på +1000 varenumre, hvilket giver anledning til et system der netop kan rumme dette.

Målet med dette projekt er, at skabe et brugervenligt og enkelt system, som medarbejderne både på lageret og på kontoret ville kunne tilgå og benytte sig af, i de forskellige arbejdsprocesser. Det skal kunne rumme et produktkatalog, varestatus, ordrestatus samt diverse gængse funktioner som der for eksempel er og kan være brug for på lageret.

2 Systembeskrivelse

I dette afsnit er overblik-systemet og dets overordnede struktur beskrevet.

Først og fremmest så er systemet tiltænkt de administrative medarbejdere samt lagermedarbejderne, da det er begge afdelinger der skal kunne tilgå de informationer, som skal være tilgængeligt i systemet. Dog er der funktioner som ikke bliver brugt af begge parter.

En lager medarbejder skal for eksempel ikke indskrive en ordre, men kun vælge en ”ledig” ordre. Til gengæld skal begge parter kunne tilgå et overblik over lagerbeholdning for et specifikt varenummer.

Systemet er i bred udstrækning tænkt som en løsning på virksomhedens nuværende system og arbejdsgange. Virksomhedens har et ønske om at skabe en digital platform, hvor alt kører via computere og håndholdte terminaler. Det vil også kunne minimere svind, fejl og mangler signifikant. Sådan et system er ekstremt omfattende og vi vil i denne proces starte med at lave en form for skelet til, hvad sådan et system også skal indeholde og kunne. Som start bliver systemet blandt andet bygget op omkring, at de administrative skal kunne danne sig et overblik over hvilke ordrer de har inde, samt hvilke der er påbegyndt. Vi vil også gerne skabe en platform, hvorpå vi kan opbevare et produktkatalog, samt en form for lagerstyring af produkter der ryger ind og ud af lageret. Ydermere vil vi skabe en oversigt over produkternes lokation på lageret.

Systemets struktur skal bygges op på en hensigtsmæssig og brugervenlig måde, da forbrugerne er en bred gruppe af medarbejdere med stor diversitet i deres erfaring med, at benytte sig af digitale systemer.

3 Information om produkter, ordrer og lokationer

I dette afsnit præsenteres modeller som udtrykker den del af virkeligheden, der skal håndteres i systemet.

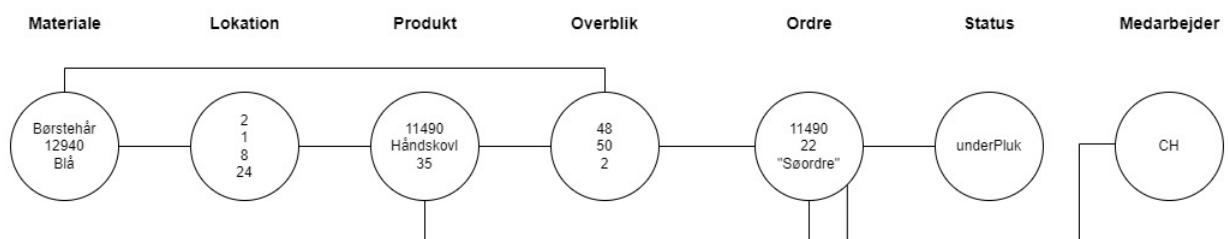
Objektmodel:

En objektmodel er en model som udtrykker ting, begreber og koncepter inden for den virksomhed, der skal udvikles et system til - modellen visualiserer objekter i den virkelige

verden, som systemet skal afspejle. I modellen (Figur 1) kan man se konkrete eksempler på de informationer, der skal behandles i systemet og hvordan de hænger sammen. Objektmodellen laves ved at identificere de nødvendige stamdata samt de objekter som relaterer til disse stamdata.

Vi har lavet et system, som skaber et overblik over produkter og ordre på FBK's lager med de nedenstående kvalitetskriterier.

- Der skal være associationer mellem objekter (streger)
- Det er ikke objekter i programmer eller database, men en afspejling af den virkelige verden
- Der skal være konkrete data i objekterne



Figur 1: Objektmodel

Domænemodel:

Efter dataene til objektmodellen er behandlet og sat sammen, bliver domænemodellen udarbejdet. Domænemodellen er et overblik over systemets objekter, inddelt i kategorier som 'klasser' og 'attributter'. Vi benytter os af de objekter fra netop objektmodellen til, at skabe domænemodellen. Hver klasse er forbundet med en anden klasse illustreret med streger og deres tilhørende multiplicitet er illustreret.

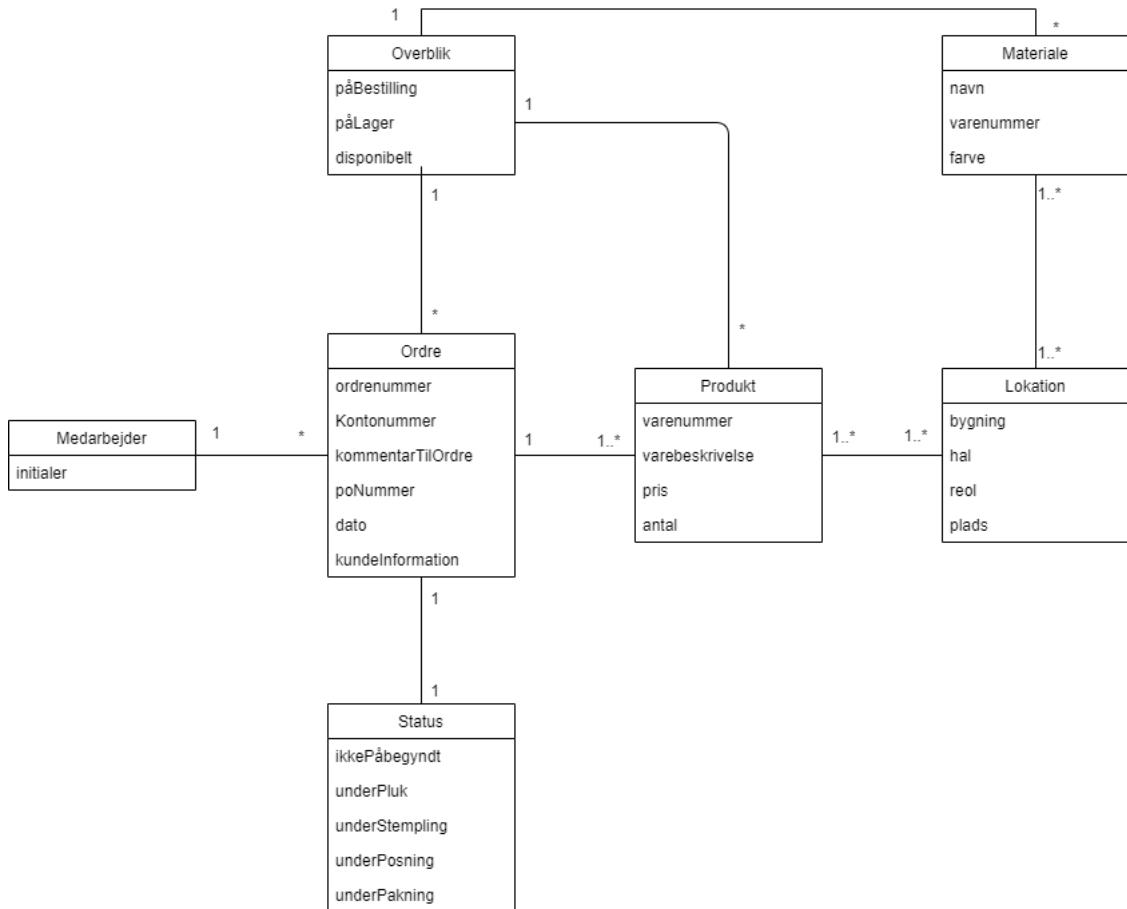
Stregerne mellem klasserne betegner deres Association med hinanden. Multipliciteten viser hvilken sammenhæng der er fra den ene klasse til den anden og omvendt.

Denne fase er den del hvor vi lærer hvordan virksomheden fungerer, hvilke arbejdsgange vi skal observere for, at kunne udvikle det system som skal hjælpe dem med deres problemstilling. Med andre ord så vil vi gerne modellere virkeligheden ned i modeller og diagrammer. På den måde skaber vi en sammenhæng mellem det fysiske arbejde og det system som skal benyttes.

Følgende kvalitetskriterier er blevet brugt til, at udforme domænemodellen.

- Klasse navne skal være i ental.
- Associationer mellem klasserne.
- Der skal være multiplicitet mellem klasserne.
- Modellen skal udfyldes til at afspejle hvert objekt i en objektmodel til en klasse.

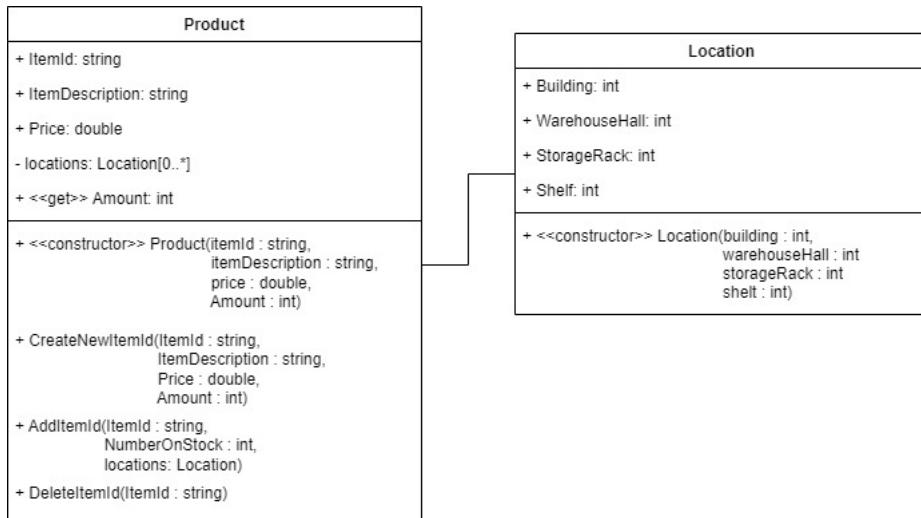
(Figur 2) viser domænemodellen med dens associationer og multipliciteter



Figur 2: Domænemodel

DCD:

DCD (Designklasse diagram) for vores 'Product' og 'Location' klasse (Figur 3). Disse er lavet ud fra vores domænemodel og er et diagram som illustrerer specifikationerne i software klassen, interface og deres attributter. DCD'en er ikke kun et visuelt diagram, den er også et værktøj og bruges som byggesten til selve kode-delen. Her arbejder vi ud fra vores forskellige DCD'er. Således kan vi sikre os, at vi benytter samme metode- og attributnavne hele vejen rundt, uanset hvem som skulle arbejde på de forskellige dele af systemet. Dette er også en af de kvalitetskriterier som vi diskuterede i opstarten, da vi ville stræbe efter en tydelig sporbarhed igennem alle modeller og diagrammer.



Figur 3: Designklasse

Nedenstående model (Figur 4), viser de to ovenstående designklasser samt en oversat ordliste over de attributter som vi benytter i softwaren. Denne model giver overblik i, hvad der i vores domænemodel bliver oversat til hvad. Dette gør vi, da vi i vores software udelukkende benytter engelske navne, og ved hjælp af denne ordliste giver virksomheden en hjælpende hånd til at forstå de forskellige navngivninger. En anden årsag til at vi også benytter os af engelske navne og termer i designklassen og i selve kode-delen er, at det giver bedst mening i forhold til, at de indbyggede biblioteker også er på engelsk. Så for at være konsistent og have den mest optimale struktur og tråd igennem koden, gør vi således. Det er i øvrigt også praktisk, hvis udenlandske programmøre skulle overtage eller videreudvikle på systemet.

Designklasse	Dansk	Engelsk
Product	Varenummer	ItemId
	Varebeskrivelse	ItemDescription
	Pris	Price
	Lokation	locations
	Antal	Amount
+ ItemId: string		
+ ItemDescription: string		
+ Price: double		
- locations: Location[0..*]		
+ <<get>> Amount: int		

Location	Dansk	Engelsk
Location	Bygning	Building
	Hal	WarehouseHall
	Reol	StorageRack
	Hylde	Shelf
+ Building: int		
+ WarehouseHall: int		
+ StorageRack: int		
+ Shelf: int		

Figur 4: Designklasse og Ordliste

Use case:

Der er løbende i projektet gjort brug af use-cases. Use case er en tekst der beskriver hvordan systemet agerer når en aktør benytter systemet. Nedenstående (Figure 5) use case er beskrevet i *brief* format, som er den ene ud af de tre use case detaljeringsgrader. I dette format er det beskrevet, som et enkelt afsnit, af systemet hvis alt går ideelt.

Inden use casen blev udviklet, blev der stillet krav til use casen, også kaldet kvalitetskriterier. Kvalitetskriterier er de kriterier use casen skal vurderes i forhold til. Disse nedenstående kvalitetskriterier er udviklet på baggrund af undervisnings litteratur[1] og gruppens egne interne krav til use casen.

- Alle informationer der bruges til use case skal forekomme fra domænemodellen.
- Skal beskrive hvad brugeren gør.
- Skal beskrive den funktionalitet som brugeren forventer.

I følgende use case, med kvalitetskriterierne i mente, bliver et allerede eksisterende produkt tilføjet til lageret.

Tilføj produkt

Når der er produceret et produkt, som allerede findes i produktkataloget, så finder den administrativt-ansatte produktet i systemet ved at indtaste varenummeret. Den administrativt-ansatte indtaster antallet af produkter til lageret. Systemet tilføjer antallet og opdaterer lagerbeholdningen.

Figur 5: Use-case: Tilføj Produkt

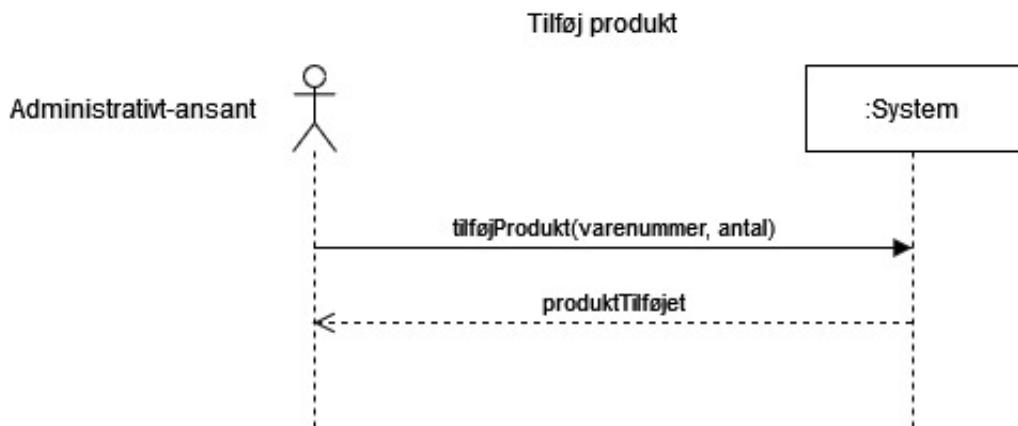
SSD:

Ud fra ovenstående use case, er der blevet udviklet nedenstående SSD (Systemsekvensdiagram) (Figur 6). Formålet med et SSD er at få defineret og identificeret affære, som systemet skal håndtere. Altså hvilket input fra brugeren systemet skal reagere på og hvilket output det skal returnere, her repræsenterer hver pil ind til systemet en systemoperation. Når der skal udvikles et SSD, kan det i visse tilfælde være nødvendigt at træffe nogle beslutninger om brugergrænsefladen, da det er konkrete input og output man har et gøre med.

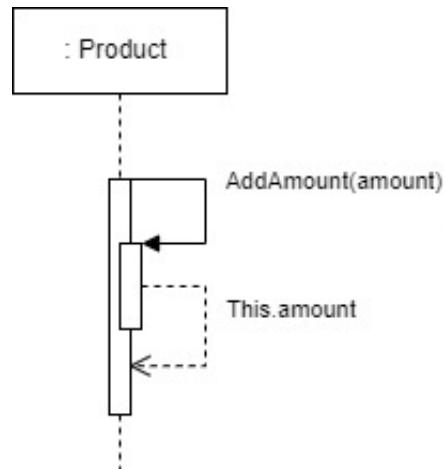
Følgende kvalitetskriterier er blevet udarbejdet til kontrol af SSD'erne.

- Udføres fra use case.
- Arver navn fra use case.
- Det skal fremgå i diagrammet hver gang der er en interaktion mellem bruger og system.
- Sekvens er udført i rigtig rækkefølge.

Den følgende SSD og SD beskriver hvordan en administrativt-ansat vil tilføje et produkt til lager.



Figur 6: SSD: Tilføj Produkt



Figur 7: SD: Tilføj Produkt

Operations kontrakt:

En OC (Operationskontrakt) er baseret på en SSD og har til formål, at beskrive en enkelt systemoperation i systemet skal gøre. Her bliver der fastgjort hvilke input der modtages og hvordan resultat har virkning på systemet, det kan ses på nedenstående (Figur 8).

Følgende kvalitetskriterier er blevet udarbejdet til kontrol af OC'erne.

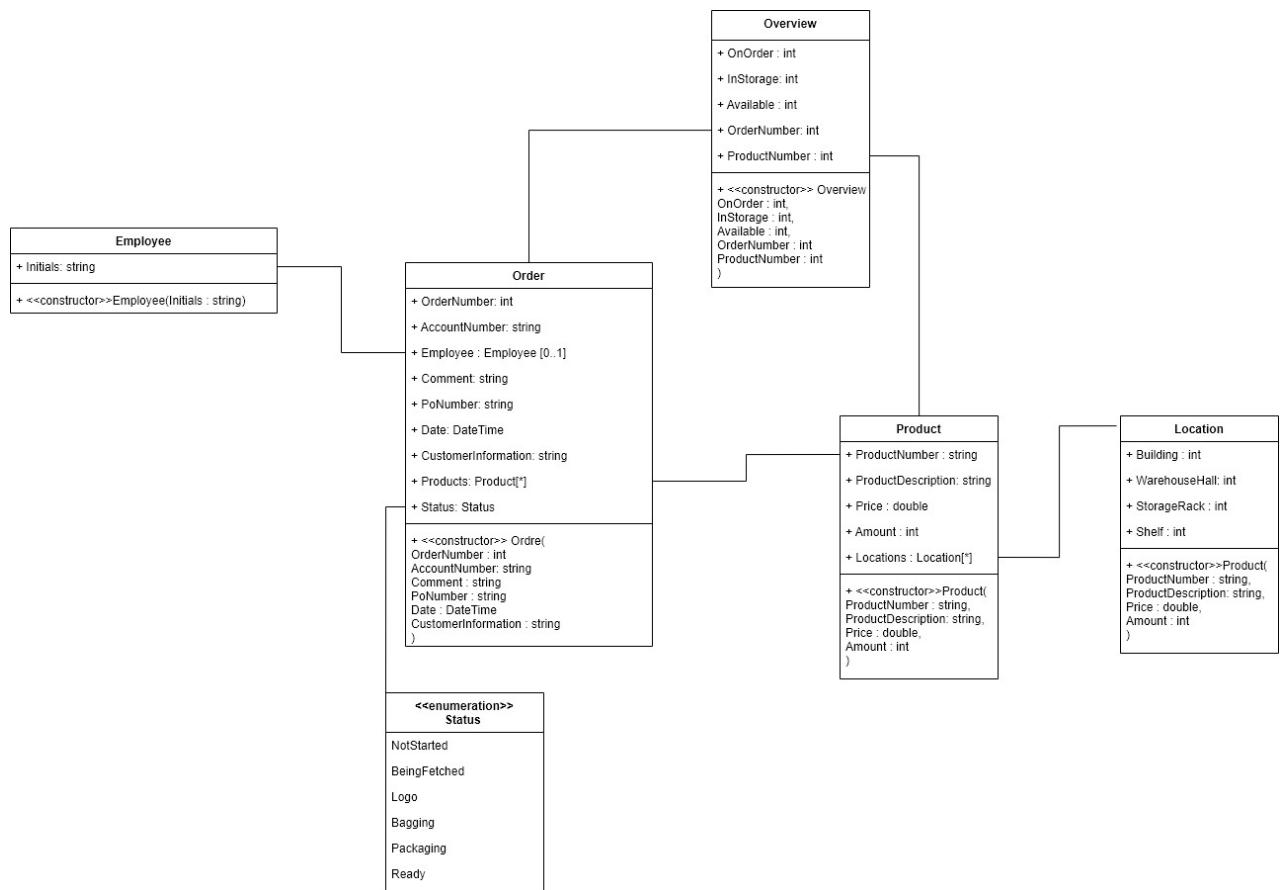
- Beskriver systemoperationer og deres input og resultater.
- Sporbarhed i forhold til navn og sprog.

- Fokus på det vigtigste element: postcondition.

Operation	tilføjProdukt(varenummer : int, antal : int)
Cross References:	Tilføj produktion
Preconditions:	<ul style="list-style-type: none"> - Et objekt type af "Produkt" eksisterer
Postconditions:	<ul style="list-style-type: none"> - Antallet af produkter tilføjes til objektet "Produkt"

Figur 8: Operationskontrakt: Tilføj Produkt

Tidligere har vi beskrevet DCD for en konkret klasse (Figur 3). Figur 9 illustrerer DCD'en for det komplette systemet.



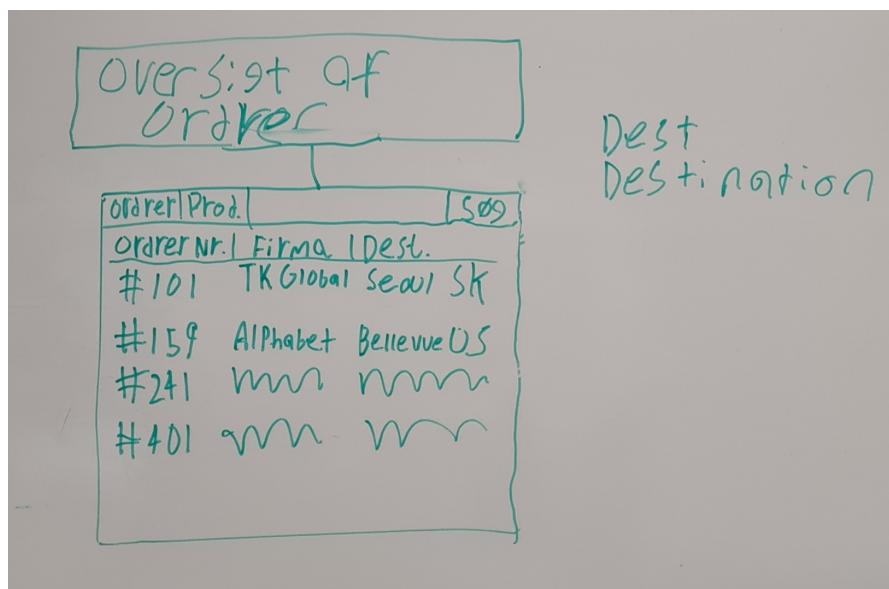
Figur 9: DCD for hele systemet

Kriterier for kvalitetskriterier: For at sikre kvaliteten til kvalitetskriterierne til de forskellige artefakter, bliver man nød til stille nogle krav til kravene. Når det handler om kriterier

af kriterier er der 3 ting der gør sig gældende, og det er de 3 K'er, Korrekt, Konsistens og Komplet. Det skal sikre at der er en rød tråd gennem alle artefakterne med klar sporbarhed. Det er dog vigtigt at pointere, at man ikke kan tjekke alting. Derfor bliver man nød til at vælge noget konkret ud og vælge det, der er mest kritisk.

4 Design

For at komme frem til et tilfredsstillende design, skal der være sat nogle krav for systemet, derfor er der blevet taget udgangs punkt i de 10 brugbarhedsheuristik[2] og C.R.A.P. design principper[3]. Systemet skal kunne give et hurtigt overblik over ordrene, samt give flere detaljer for selve ordren. Der skal være mulighed for at ændre på selve produktet, fx hvis dette produkt har fået en ny lokation, eller hvis dette produkt er udgået. Der skal være en nem og hurtig måde at finde en ordre, om dette så er en gammel eller ny ordre. Administrationen skal også have mulighed for at oprette nye ordre, samt at ændre status på den. Dette er ikke noget som en typisk lager ansat kommer ud for, da de blot skal finde de produkter som hører til ordren. Ud fra disse krav er der blevet skitseret nogle wireframes af hvordan systemet skal se ud.



Figur 10: Wireframe af ordre oversigt

I de følgende to Figurer 11 og 12 er der eksempelvis gjort brug af nummer 4 af de 10 brugbarhedsheuristiker, der handler om konsistent og standarder. Det kan ses ved henholdsvis i pilen der peger til højre i knappen i Figur 11 og luppen i søge funktionen i Figur 12, da det er det bruger har kendskab til fra andre applikationer og forventes fra applikationer der implementere samme funktionalitet.

Visning af specifik Ordre			
Ordre / Prod			
Vare Nr	Antal	Placering	Vare Beskrivelse
15150-2	24	L1.09.139	Hanger for ø 22-32mm
15151-2	60	L1.09.140	Hanger for ø 28-38mm
28243-5	12	L1.06.187 -185	Hand squeegee 300mm

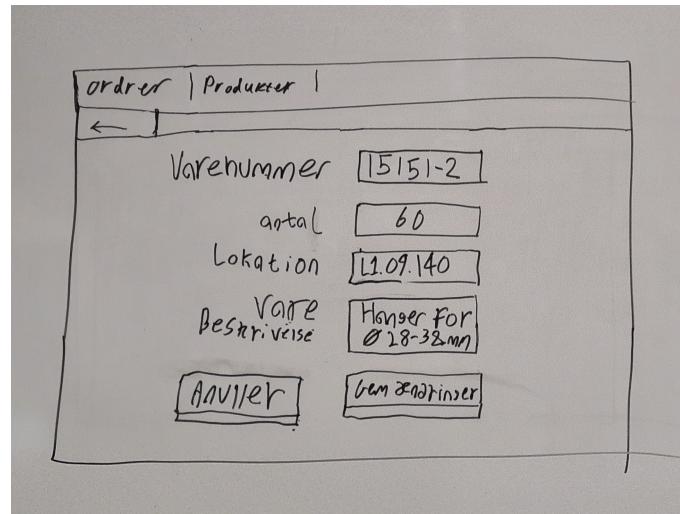
Figur 11: Wireframe af specifik ordre

OVERSigt af Produkter			
Ordre / Produkter Søg Ø			
Vare Nr	Antal	Lokation	Vare Beskrivelse
15150-2	560	L1.09.139	Hanger for ø 22-32mm
15151-2	469	L1.09.140	Hanger for ø 28-38mm
28243-5	420	L1.06.187 -185	Hand squeegee 300mm

Figur 12: Wireframe af oversigt af produkter

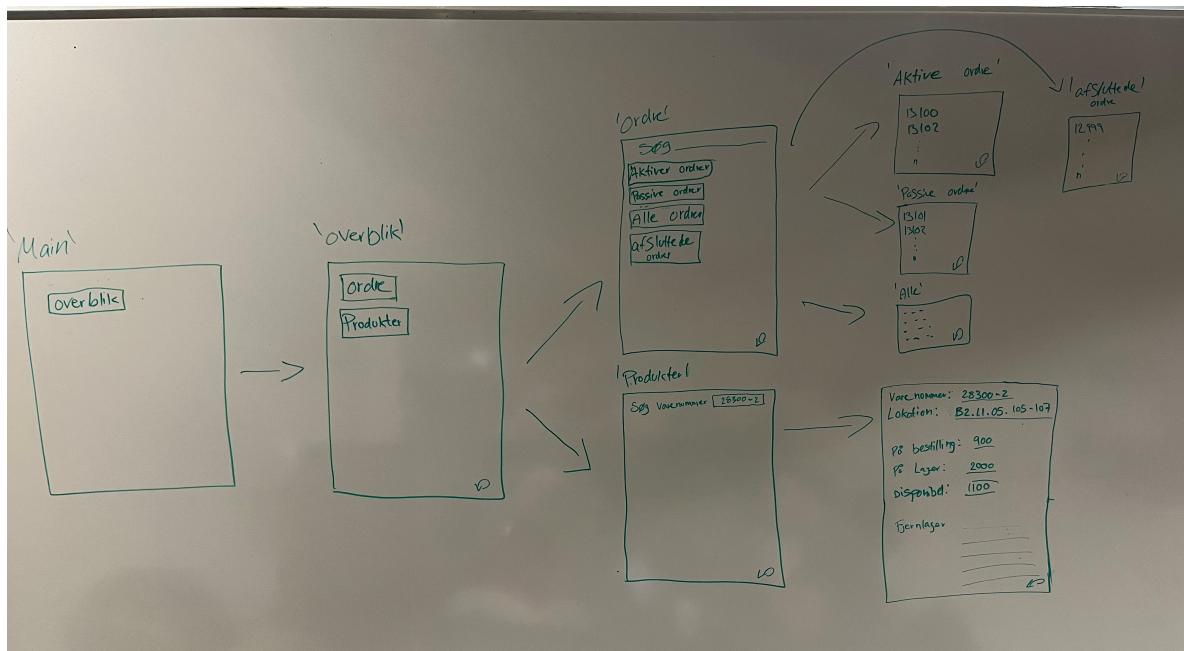
I Figur 13 er der gjort brug af nummer 6 af de 10 brugbarhedsheuristiker, der omhandler at minimere brugerens hukommelsesbelastning. I stedet for at bruger skal kunne huske

alle de relevante informationer om produktet, vil de persistere for nemmere genkendelse. Brugeren har ikke brug for at huske f.eks. hvad varenummeret var inden de skiftede vindue.



Figur 13: Wireframe af rediger specifikt produkt

Der blev også lavet et alternativ til hvordan systemet kunne se ud. I nedenstående Figur 14, er der lavet et userflow baseret på wireframes.



Figur 14: Wireframe af brugerflade

5 Implementering

Windows, pages and user controls

I implementeringen vil vi komme ind på samarbejdet af vores windows, pages og user controls, hvordan deres sammensætning af systemet fungere. Der kommer en vejledning til hvordan vores interface fungere, samt en begrundelse på det afsluttende interface design.

Interface

Interface har udviklet sig ud fra designet, der er stadig holdt til start princippet. Der er nogle af hoved metoderne, som ikke er implementeret endnu, men vil komme i det færdige udviklings system. Disse metoder som følgende: søge funktion samt afslut og åbne ordre. Ordre oversigt samt oprette ordre er ikke færdig udviklet, men er en af de ting som skal være oppe at køre, inden udgivelse af systemet. Tilrettelser og tilføjelser til systemet, kan altid tilføjes på et senere tidspunkt.

Oversigt

Systemet er bygget op på den måde så, der er et nemt overblik over de muligheder man har. Om du har åbnet et produkt, eller skal lave et nyt produkt, er der altid en direkte metode at gå over til hvor du vil hen. Under oversigt er der også mulighed få at gå ind at redigere, hvis der er ændring i ordrenummer eller andet i forhold til position.

Opret

Når der skal oprettes ordre eller produkter, skal der noteres de informationer der har noget at gøre med Fyens Børste- og Kostefabrik. Det som der har noget med kunden, er ikke informationer som den ansatte skal skrive ind. De vigtigste informationer er produkt, antal samt lokation for en ordre, ved vare er det lokation, varenummer og antal på lager. Antal på lager er vigtigt for lagermedarbejderne, da de skal sørge for at få fyldt lageret op, når der begynder at vare lavt lager inventar.

User controls

User controls er de elementer de visuelle elementer som vi skulle genbruge en del. Eksempelvis under produkternes overveiw, benytter vi et user control der hedder product listing.

```

namespace GettingReal.layout.components
{
    /// <summary>
    /// Interaction logic for ProductListing.xaml
    /// </summary>

    public partial class ProductListing : UserControl
    {
        public ProductListing(string itemId, string amount,
        string price, int count)
        {
            Brush c = (count % 2 == 0) ? Brushes.LightGray :
            Brushes.Moccasin;

            if (count == -1)
                c = Brushes.White;
            Trace.WriteLine(String.Format("{0}\n{1}\n{2}", itemId,
                amount, price));
            InitializeComponent();
            ItemId.Text = itemId;
            Amount.Text = Convert.ToString(amount);
            Price.Text = price;

            this.Background = c;
            this.Resources["ItemId"] = itemId;
        }
    }
}

```

```
}
```

Ovenstående user control bliver instantieret, når brugeren klikker tilføj på ”Create New Product”. Logic filen der ligger bagved .xaml filen, kalder så på ”ProductController.AddProduct” med de intastede værdier. Denne controller kalder så til aller sidst på MainControlleren. MainControlleren kommunikerer så med ProductOverview, og således skabes der et nyt produkt i den eksisterende liste.

Window and Page navigation

Hele applikationen er i bund og grund et window med en masse forskellige pages på en Frame. Når vi navigerer til at lave et nyt produkt, eller navigerer til oversigten over produkter, så er det eneste der reelt sker, er at Framen skifter page. Fra man starter programmet, *MainWindow()* bliver begge repositories initialiserer (Order og Product). Efter dette, så bliver de tre sub-controllere initialiseret. ProductController, ProductOverviewController og OrderController (OrderController blev desværre ikke implementeret grundet tidsbegrænsninger. Efter dette, så bliver alle pages initialiseret og bliver gemt i MainControlleren. Nu er programmet klar. Navigations knapperne i toppen af programmet fungerer nu på den måde at når du klikker på dem, så bliver der kaldt en metode på MainWindow, som kalder videre på MainControlleren, som så skifter framet til den page man har valgt. Dette bliver gjort ved hjælp af DataContext property på vores frame, som vi sætter til en af vores pages som er klar til brug. Dette gør at når vi skifter fra den ene side til den anden, så bliver alt input gemt, og er klar til at blive redigeret videre på når man skifter tilbage til samme page. Det gør dertil også applikationen hurtigere end den første metode vi valgte at bruge, som var at sætte source på framen til URI'en til de forskellige pages. Årsagen til det er hurtigere er så at hver gang man skifter side, så bliver der initialiseret en ny page, som så bliver smidt ind, istedet for at vi allerede har en page klar som bare bliver smidt ind.

6 Business case

Når vi nu har beskrevet ideerne bag udviklings- og designprocessen af systemet, vil vi definere udbyttet virksomheden kan forvente i form af, både potentielle fordele og ulemper.

Beskrivelse af problemområdet:

Forretningsmæssig baggrund:

FBK er en succesfuld produktionsvirksomhed, som producerer og sælger et bredt udvalg af industriprodukter som blandt andet fejekoste, gulvskrubber og ophængssystemer til virksomheder på alle jordens kontinenter.

Med en produktionsafdeling som til dagligt bygger produkter med et udvalg af +1000 varumønstre, er det essentielt for virksomheden at lageret både er korrekt opstillet og sorteret.

Forretningsmæssig problemstilling:

Med et manuelt papirsystem i forhold til lager-styring vil der hurtigt kunne ske tastefejl i forhold til dokumentation. Det er sket for lagermedarbejderne som plukker de forskellige byggematerialer til produktionen, at de nødvendige materialer ikke er på deres korrekten plads eller at produktionsmedarbejdere, ikke har registreret afhentning af materialer.

Det vil i sidste ende skabe unødvendigt økonomisk svind, hvis FBK bliver nødt til at, at bestille unødvendige varer hjem, eller lagermedarbejdere skal bruge unødvendig tid på at finde varer de ikke nødvendigvis har på lager.

Interessenter:

Lageransvarlig og Indkøbsmedarbejdere vil være, de to primære brugere af systemet. Den lageransvarlige vil nemmere kunne skabe sig et overblik over hvordan lagerbeholdningen forløber og indkøbschefen vil nemmere kunne danne sig et faktuelt billede af de nødvendige indkøb.

Produktionslederen og Salgsmedarbejdere ligeledes også kunne anvende systemet til henholdsvis skabe et overblik over mulige produkter at producere og hvor længe det vil tage

at få et produkt fra bestilling klar til salg.

- Systemets brugere
- Lageransvarlig
- Indkøbsmedarbejdere
- Produktionsansvarlig
- Salgsmedarbejdere

Scenarie oversigt:

0-scenariet: Den nuværende status-quo fortsætter.

- Fordeler: Der vil ikke skulle investeres i et nyt it-system.

Nye it-systemer er en stor økonomisk investering og kan tage lang tid, at få ordenligt implementeret. Medarbejdere skal ikke omlægge deres arbejdsgang, men eksisterende systemer skal blot strammes op.

- Ulemper: De samme fejl som opstår nu, vil ikke blive løst.

At stramme op på allerede eksisterende arbejdsrutiner, er en stor tidsmæssig investering. Det kræver ekstra tid og arbejde fra personalet, som ville kunne udnyttes bedre i produktion frem for renskrivning. Hvis virksomheden udvikles og personalet for mere travlt, vil lagerdokumentationen hurtigt blive nedprioriteret.

1-Scenariet: Forenkle lagerstyringsprocessen og effektivisere overblikket.

- Fordeler: Et digitalt lagersystem fylder mindre end et fysisk og det vil blive markant hurtigere at danne overblik over lagerbeholdningen, for lager- og produktionschefen. Der kan samtidigt også altid udvikles videre på et it-system, hvis der senere hen, bliver brug for mere funktionalitet.
Lagerarbejdere kan, med funktionaliteten ”lokation”, hurtigt finde de efterspurgte materialer som skal bruges til en bestilling.
Det er samtidigt også nemmere at lave backup og sikkerhedskopier af et digitalt lagersystem, frem for fysiske papirformler.

- Ulemper: Nye it-systemer er både en dyr økonomisk- og tidsinvestering. Der skal bruges tid på at oplære personalet i at bruge det nye system og der kan gå lang tid før det bliver rutine som sidder på rygraden.

2-Scenariet: Effektiviser den nuværende arbejdsgang med ekstra kontrol.

- fordele: Der skal ikke investeres i et nyt it-system.

- Ulemper: Det kommer til at koste arbejdstimer at dobbelttjekke andres arbejde og skrive på papir og Det kan være svært at læse andres håndskrift.

1-Scenariet:

Vi anbefaler at investere i det nye digitale lagersystem, som både Indkøb- Lager- og produktions- og salgsafdelingen kan få glæde og gavn af, med udgangspunkt i et bredere overblik og nemmere kommunikation mellem afdelingernes arbejdsopgaver.

Formål:

Formålene ved Det nye lagersystem er:

- 1:** Forøge hastigheden ved overbliks-dannelse af lagerholdning i forhold til antal og lokation af materialer.
- 2:** Give øre det hurtigere for lagerpersonalet at finde og plukke materialer som skal sendes til produktionsafdelingen.
- 3:** Give indkøbsafdelingen nemmere ved, præcist at kunne se hvilke vare som skal bestilles hjem.

Forretningsmæssig løsningsbeskrivelser:

Lagerstyringssystemet skal sikre at lagerafdelingen altid kan give et præcist og nøjagtigt svar til produktionsafdelingen angående hvad lager beholdningen er, hvad der er tilgængeligt på lageret og hvad der er bestilt hjem af indkøbsafdelingen.

Det forventes at en computer skal opsættes ved varemøntagelsen på lageret, så der kan registreres nye vare, som kan tælles ind i beholdningen. Samtidigt skal produktionsafdelingen kunne vælge materialer som skal trækkes fra lagerbeholdningen når de bestiller fra lageret. Når der er fjernet et bestemt antal fra lagerbeholdningen skal indkøbsafdelingen have besked fra systemet om at nye skal bestilles hjem.

it-mæssig løsningsforslag: Lagerstyringssystemet udvikles i .NET C# og skal kunne køres på PC'er.

Gevinster:

Interne serviceforbedringer: Både Lager-, produktions- og indkøbsafdelingen vil få nemmere adgang til korrekt og opdateret information på kortere tid.

7 Proces

Sprint 1 - 2 - 3

Før vi kunne starte arbejdet med de første artefakter som objekt- og domænemodellen, blev vi nødt til at se på problemstillingen og den case vi blev stillet af virksomheden. For at løse problemerne på den bedst mulige måde og samtidig komme i mål med et system som opfylder de ønsker og behov der er stillet af virksomheden, er vi nødsaget til at forstå arbejdsgangene fra start til slut.

I begyndelsen havde vi meget snak frem og tilbage omkring de forskellige arbejdsgange, netop for at kunne forstå hvad der egentlig skete og hvilke behov der var. Efter at være kommet til en fælles forståelse kunne vi indlede de første skridt med at modellere oplysningerne ned i modeller.

Som en del af sprint 1 [4] planlagde vi hvordan vi løbende ville arbejde på flere ting ad gangen, således at vi ikke skulle stå til sidst og evaluere og tænke tilbage på hvilke tanker og overvejelser vi gjorde os i starten. Vi lavede derfor nogle små delmål undervejs i de 3 sprints, som skulle være en hjælp til at holde os nogenlunde på sporet. Vi måtte dog erkende at ikke alle delmål blev nået eller overholdt hvilket vores retrospektiv møder heller ikke gjorde. Som resultat af dette evaluerede vi hvad der gik galt, hvorfor det gik galt og hvad der eventuelt skulle laves om til de efterfølgende sprints.

Et eksempel på hvordan vi har prioriteret arbejdet i de forskellige sprints, kan vises via en product backlog.[4]

Product Backlog

I forbindelse med planlægning af sprint 1 og 2, lavede vi en product backlog med udvalgte items som vi ville arbejde i dybden med - både dens tilhørende artefakter og selve implementeringsdelen.

Figur 15 viser denne product backlog med henholdsvis prioritering, hvilket sprint de skal udarbejdes i, samt deres forventede sværhedsgrad.

Hvad er en product backlog så? Kort sagt er det en form for ordnet liste med prioriteringer, for hvad der skal til for at komme i mål med produktet. I dette tilfælde ville vi gerne starte med at kunne oprette og tilføje produkter, da det er essentielt at have produkter registreret i systemet, for at kunne danne sig et overblik over disse efterfølgende.

De forskellige items er tilføjet i samarbejde med vores product owner, som har kunne give information på hvad der er mest nødvendigt for virksomheden og brugerne af programmet. På figuren er der dog sneget sig en fejl ind, de ikke står i den rækkefølge de burde. De vigtigste og højst prioriteret items skal fremgå øverst og dernæst de efterfølgende og så videre. ”Produkt-overblik” burde derfor stå nederst i denne figur.

PRODUCT BACKLOG						
User Story ID	User Story	Priority	Sprint	Task owner	Estimated effort	
US001	Produkt-overblik. Som lageransat kan jeg få et overblik over en ordres produkter som er på bestilling, deres lagerstatus og eventuelt den disponibele andel af produkter.	3	2		High	
US002	Opret produkt Som administrativt-ansat kan jeg oprette et produkt ind i lager. Produktet har følgende data: varenummer, varebeskrivelse, pris, placering (lokation)	1	1		Medium	
US003	Tilføj produkt Som administrativt-ansat kan jeg indtaste et produkt til lager, som bliver tilføjet til det overordnet lager. Produkt har følgende data: varenummer, antal	1	1		Low	

Figur 15: Product Backlog

Evaluering

Efter endt sprint gennemgik vi det udarbejdet materiale og lagde en plan for den kommende sprint. Vi gennemgik især de ting som ikke var gået som planlagt. En del af det at lave et projekt som dette, er også at få et indblik i hvordan virkeligheden hænger sammen, i forhold til at man ikke altid når det man gerne vil til deadline.

Til trods for det, haft vi haft nogle udfordringer i teamet som heller ikke har hjulpet os i den rigtige retning. Dette er noget vi har været opmærksomme på og også diskuteret, om end det har været lidt en bremseklos for fremdriften af projektet. Dette føler vi dog ikke som spildt arbejde eller ser det som en negativ ting. Tværtimod prøver vi at se det som en del af den proces vi er i, og bruger den til at styrke samarbejdet og lærer af de fejl vi har begået i dette forløb.

8 Konklusion

Processen i projektarbejdet har lært gruppen, hvor vigtigt det er at opretholde åben kommunikation mellem forskellige afdelinger af projektets retning, samt planlægning og opfølgning på vores SCRUM møder.

Gruppen kom godt fra start og fik -i samarbejde med både undervisere og FBK- fundet frem til et lagerstyring system.

I Sprint 1, fik vi hurtigt udviklet projektets første artefakter, fik holdt møder på for at få aftalt og uddelegeret arbejdsopgaver.

I Sprint 2, efter at have startet på koden og snakket med FBK, blev vi hurtigt opmærksom på forskellige aspekter ved de grundlæggende artefakter, skulle revurderes. Vi begyndte at opleve udfordringer, ved opfølgning i processen, hvilket resulterede i manglende forståelse af projektets retning.

Efter opsamling, genovervejelse af projektets retning og forventningsafstemmelse i forhold til arbejdet, fik vi rettet projektet ind og Sprint 3 forløb effektivt.

På trods af et mere effektiv sprint 3 er vi ikke kommet i mål med et færdigt produkt. Grundet det efterslæb som vi havde fra sprint 1 og 2 sad vi med en for stor arbejdsbyrde til sidst. Vi har dog fået en god læring i hvor vigtigt samarbejde er for sådan et projekt og hvorfor planlægning samt en kontinuerlig opfølgning er så vigtig en del.

Litteratur

- [1] Craig Larman. "Applying UML and Patterns". I: 2005. Kap. 6, s. 61–101.
- [2] Jakob Nielson. *10 Usability Heuristics for User Interface Design*. URL: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- [3] Nitin Deshdeep. *How to Use C.R.A.P. Design links with Principles for Better UX?* URL: <https://vwo.com/blog/crap-design-principles/>.
- [4] Michael James Luke Walter. *Scrum Reference Card*. URL: <https://scrumreferencecard.com/ScrumReferenceCard.pdf>.