



POWERBANKEN

1. Års eksamensprojekt

Jakob Vingaard Petersen

Rasmus Bak Petersen

Daniel Stuhr Petersen

Jacob Lau Petersen

Simon Egebjerg Jensen

Gruppe C2
OEADM17EDC

Indhold

Læsevejledning	5
Præsentation af Powerbanken.dk.....	5
Forretningsmodel.....	5
Nøglepartnere	5
Nøgleaktiviteter	6
Nøgleressourcer	6
Value-proposition	6
Distributionskanaler	6
Kunderelationer.....	7
Kundesegmenter	7
Business case - Powerbanken.dk	7
Formål med projektet.....	7
Nuværende situation.....	7
Nuværende Arbejdsprocesser i virksomheden	7
Fremtidig situation efter indførelse af løsningen	8
Alternative løsningsforslag	9
Scope og afgrænsninger	10
Scope	10
Afgrænsninger:	10
Mål og succeskriterier	11
Økonomiske gevinster	11
Ikke økonomiske gevinster	12
Key Performance Indicators	13
Interessenter	14
Handlingsplan	14
Kommunikationsplan.....	15
Problemformulering.....	16
Brugervejledning til systemet	16
Dokumentation af bestillingsdatoer.....	19
Modellering af systemet.....	20
Use-cases	20
Vurderingskriterier	20

Udformning af use-cases	20
Refleksion over use-cases.....	20
Use case - UserGetsNotificationToOrderProduct.....	21
Objektmodel	21
Opbygning af objektmodellen	21
Domænemodel	22
Vurderingskriterier:	22
Refleksion	23
SSD - UserGetsNotificationToOrderProduct.....	23
SOC - UserGetsNotificationToOrderProduct	24
SD – UserGetsNotificationToOrderProduct.....	25
Refleksion af SD	25
Kode - UserGetsNotificationToOrderProduct	26
Arbejdsprocessen af kildekoden	26
Design Class Diagram.....	27
Valg af arkitektur	29
MVVM.....	29
Domænelag	29
Viewmodel.....	29
View	30
Lagdeling.....	31
Testing	31
Databaseopsætning.....	32
Valg af database:	32
Normalformer.....	32
Versionsstyring	33
Objektorientert programmering	33
Polymorfi	33
Brug af Threads.....	34
Teknikken bag tråde	35
Valutaomregner.....	36
Grafisk visualisation.....	37
Projektstyring	37

Product backlog	37
Product backlog items	38
Udvikling af product backlog	39
Framework.....	39
Vores brug af Scrum	40
Burndown-chart.....	40
Refleksion over burndown-chart.....	41
Roller.....	41
Risici	42
Dokumentation af projektarbejdet	44
Eksperimenter	44
Kommunikation i gruppen	44
Resultat af eksperiment:	44
Brugervenlighedstest.....	45
Resultat af brugervenlighedstest:	45
Fremtid efter overdragelsen af projektet	46
Handlingsplan for videreudvikling.....	46
Konklusion.....	47
Litteraturliste.....	48
Bilag	49
Bilag 1 – Usecases.....	49
Bilag 2 – SSD’er	50
Bilag 3 SD	52
Bilag 4 - SOC.....	55
Bilag 5 - Product backlog item).....	55
Bilag 6 - Innovations metoder	56
Bilag 7 – KPI’er	56
Bilag 8 – Brugervenlighedstest	58
Testplan for brugervenlighedstest	58
Brugervenlighedstest – Resultater og iagttagelser	60
Bilag 9 - Handlingsplan	61
Bilag 10 – kommunikationsplan	63
Bilag 11 - Scrumboard	64

Bilag 12 - Notifikationsvindue - brugervejledning	65
Bilag 13 – Gruppe Kontrakt	66
Bilag 14 – Projektlog	67

Læsevejledning

Vi har valgt at dele vores rapport op i 4 dele. De første sider af rapporten kommer til at handle om den virksomhed, vi har arbejdet med gennem vores projekt og hvordan den hænger sammen. Den anden del kommer ind på den business case, vi har udarbejdet. Disse dele af rapporten skulle gerne give et indblik i hvilken virksomhed, vi har haft kontakt med, samt hvad vi har arbejdet med de sidste 7 uger. Herefter kommer vi til delen af rapporten, der omhandler hvordan vi har valgt at opbygge og modellere vores system. Til sidst beskriver vi den arbejdsproces, vi har haft, hvordan gruppearbejdet har været, og hvilke tanker vi har gjort os undervejs i projektet.

Præsentation af Powerbanken.dk

Powerbanken.dk er et iværksætterfirma på Fyn og er bosat her i Odense. Det er en virksomhed opstartet af Jacob Oppenheim og Christoffer Nielsen, som også er vores PO'er. I alt er de 4 medarbejdere i virksomheden. De sælger primært powerbanks, men også både kabler, led-pærer og genopladelige batterier. Dette gør de gennem deres webshop www.Powerbanken.dk, hvor de også har en supportside, hvor eventuelle kunder kan komme med spørgsmål eller kommentarer til dem. De benytter sig meget af sociale medier til at brande sig selv og skaffe nye kunder. De har bl.a. også en blog på deres hjemmeside, hvor de skriver om diverse tech, som de også sælger.

Forretningsmodel

Forretningsmodellen giver et indblik i, hvordan Powerbanken.dk skaber værdi og er opbygget. Derudover kommer vi også ind på selve strategien for virksomheden, og hvilke kundemålgrupper de retter deres produkter efter. Nogle af modellens punkter, såsom økonomi og omkostningsstruktur, er blevet fravalgt, da vi ikke har fået nok information fra Powerbanken om deres økonomi.

Nøglepartnere

Powerbanken.dk er en del af en større gruppe, som involverer BORN Gruppen og BORN Nordic Aps. Chefen for Powerbanken.dk er også medejer af førnævnte virksomheder, så de arbejder meget sammen både med logistisk hjælp og administrativt arbejde. Derudover benytter de sig også af et system til deres webhandel der hedder Magento, som de bruger til at få et overblik over salg på hjemmesiden.

Nøgleaktiviteter

Powerbanken.dk's primære indtægtskilder er deres salg af forskellige powerbanks, som også er deres primære beskæftigelse. De holder sig udelukkende til salg og service herom, og efterlader service og IT-support til andre, da det ikke er inden for deres ekspertiseområde. Alt salg foregår via deres hjemmeside Powerbanken.dk, hvor de især tilbyder forskellige tilbud i primært sommer- og vintersæson.

Nøgleressourcer

Deres mest solgte produkter er af deres eget mærke "Greylime", som de både promoverer og markedsfører mere end andre produkter. Dette er en fysisk ressource, som Powerbanken vægter meget højt i deres forretningsmodel. Derudover ejer de også et par andre små virksomheder, som vi tidligere har beskrevet under nøglepartnere, dette give en finansiell fordel for Powerbanken.dk. Herudover har de nogle gode menneskelige ressourcer, ville vi mene da teamet både er troværdige og motiverede. Endvidere er de også medejere af andre succesfulde firmaer, derfor kan vi kun konkludere, de har stærke menneskelige ressourcer.

Deres immaterielle ressourcer dækker over registrerede varemærker, såsom Anker, dette giver kunderne et indtryk af en reel webhandel, som sælger kendte produkter. Derudover har de som sagt også deres eget mærke, som kun de sælger; Greylime. Dette produkt er også meget populært hos kunderne.

Value-proposition

Det primære fokus og de mest solgte varer hos Powerbanken.dk er powerbanks, men de tilbyder også meget andet. De tilbyder både kabler, mobilhøjtalere, batterier og andet mobiltilbehør, som sælges direkte fra deres webshop. Virksomheden skaber derved værdi ved at være et samlingspunkt indenfor mobiltilbehør, da de har samlet hundredvis af forskellige produkter og en håndfuld brands under én hjemmeside

Distributionskanaler

Powerbanken.dk har en webchat på deres egen hjemmeside, hvorigennem kunder kan kontakte virksomheden, såfremt de måtte have spørgsmål vedr. produkter, levering med videre. På hjemmesiden finder man også en blog, hvor de engang imellem laver opslag omhandlende nye produkter i deres sortiment. Endvidere er de også aktive på sociale medier såsom Facebook og YouTube. På Youtube har de markedsført sig gennem deres egne videoreklamer men også gennem kendte youtubere, som de har fået til at lave giveaways eller anmeldelser af deres produkter.

De benytter sig derfor af mange online distributionskanaler for at nå deres kunder, hvilket må siges at være blandt de ret almindelige metoder for en virksomhed, som alene driver en webshop, og ikke er fysisk tilgængelig for potentielle kunder.

Kunderelationer

Der eksisterer en hjælpe-funktion på siden. Den fungerer som en relation til kunden, da det er Jacob(PO) selv, der sidder og besvarer disse spørgsmål fra potentielle kunder. Derudover er de også meget tilgængelige på sociale medier, såsom Facebook og Instagram, hvor de opfordrer kunder til at komme med ris og ros til dem. Endvidere har de også rigtig gode anmeldelser på Trustpilot.dk, som viser den gode kundeservice, de ønsker at leve op til.

Kundesegmenter

Powerbanken.dk henvender sig primært til privatpersoner, som mangler powerbanks til smartphones eller andre mobile enheder. I mange tilfælde vil det være forholdsvis unge mennesker, som bruger deres smartphone til tunge ting såsom spil, film og serier, der sluger meget strøm. Det kan dog også være ældre, der skal ud at rejse, på festival eller noget helt tredje, hvor de ikke har adgang til stikkontakter regelmæssigt. Potentielle kunder kan også være folk som mangler tilbehør til deres mobiltelefoner, som fx covers, høretelefoner eller små Bluetooth-højttalere.

Business case - Powerbanken.dk

Formål med projektet

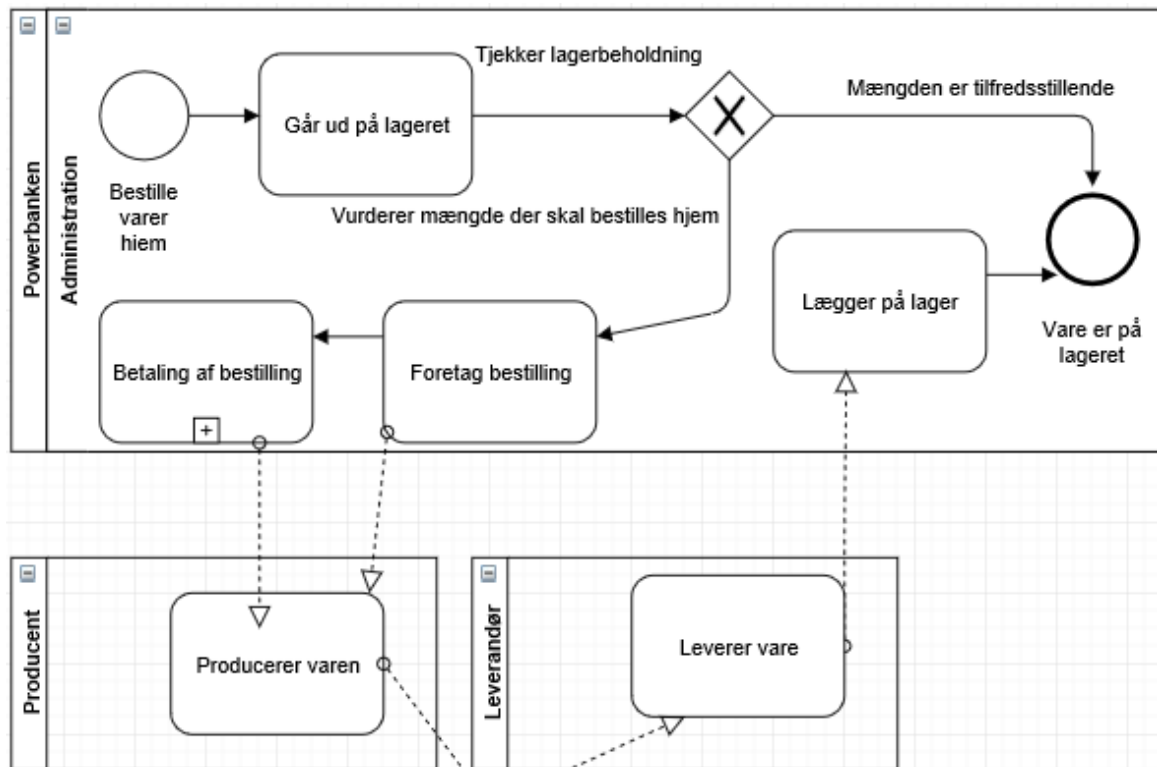
Nuværende situation

På nuværende tidspunkt går en medarbejder ud på lageret og optæller manuelt mængden af varer. Derefter kommer medarbejderen med et kvalificeret gæt på, hvor mange varer han kommer til at sælge i en given kommende periode ud fra tidligere års erfaringer. Dette kan også gøre det svært at udregne, hvor meget sæsonerne varierer fra år til år. Powerbanken.dk har også en forventet vækst af salg fra år til år, som også skal tages højde for. Dette gør det svært for Powerbanken.dk at få et overblik over hvor mange - og hvor ofte de skal bestille varer hjem.

Nuværende Arbejdsprocesser i virksomheden

Vi vil gerne visualisere den proces som Powerbanken.dk's medarbejdere foretager sig, når de skal bestille og modtage varer fra en producent. I vores model herunder ses også en subproces, som vi har kaldt "*Betaling af bestilling*". Denne har vi lavet som en subproces, da der sker flere handlinger

i den, som dog ikke er relevant for vores system. Derudover har Powerbanken.dk ikke været tilbøjelige til at give os adgang til deres betalingsprocesser. Dog er vi klar over den eksisterer og har den vist i processen. *“Betalning af bestilling”* går også kun fra Powerbanken.dk til producenten, da PO har forklaret, at når de betaler for en ordre, så vil de få fragt inkluderet i prisen, derfor betaler de kun producenten.

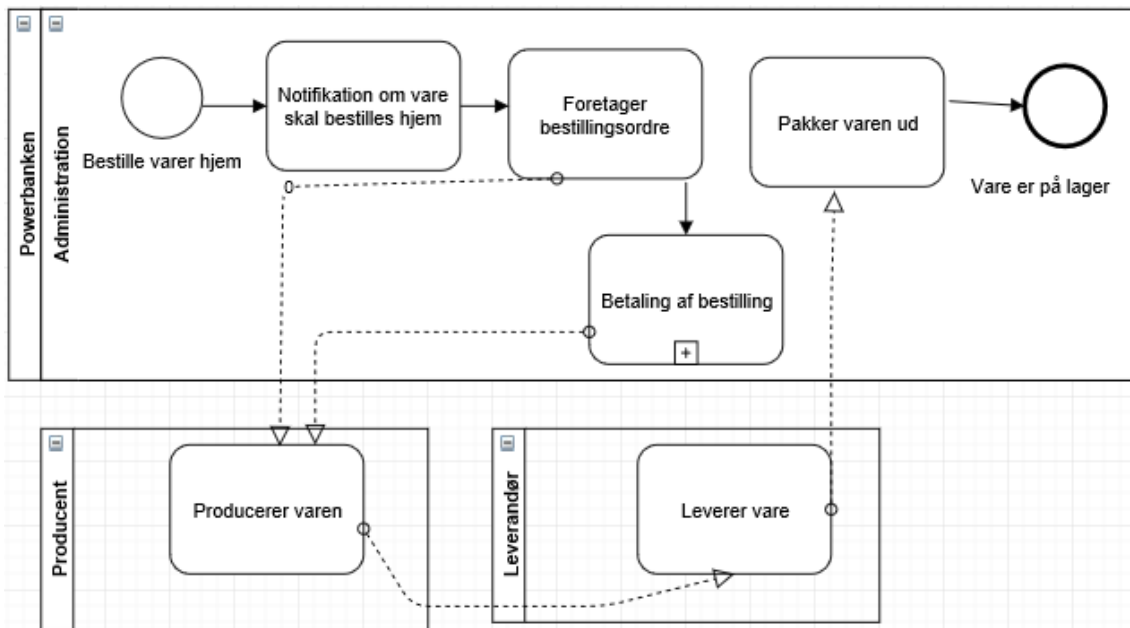


Fremtidig situation efter indførelse af løsningen

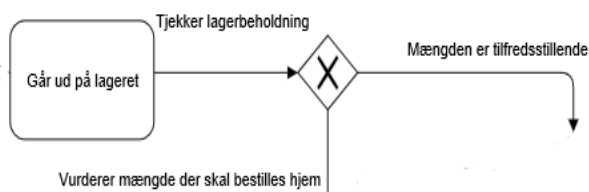
Efter implementeringen af vores løsning vil Jacob og Christoffer kunne gå ind i systemet og se hvor mange varer, samt hvilke varer de skal bestille og hvornår de senest skal bestille dem hjem for ikke at løbe tør. Derudover skal brugeren også selv have mulighed for at tage forhold til vækst i virksomheden, så hvis de har en vækst på 20% over året, på en given vare, skal systemet tage det med i sin beregning. Systemet skal også tage højde for høj- og lavsæsoner.

Systemet vil ud fra de foreslåede datoer kunne oprette ordrer til de forskellige leverandører, og de vil hurtigt kunne godkende og afsende dem.

Endvidere skal systemet indeholde en graf over totalt salg over et år, men også for specifikke varer over en given periode, brugeren selv kan vælge.



Herunder vises grafisk processen, som bliver afskåret:



Ud fra den nuværende BPMN og processen, som den ser ud nu, har vi også udarbejdet processen, efter vores system er blevet implementeret, og hvilke arbejdsprocesser, der bliver overflødige. PO vil blive sparet for turen ud på lageret, og skal ikke længere vurdere, hvor mange varer, der skal bestilles hjem. Derfor vil vi mene, han både sparer tid på optælling, men han får også en mere præcis bestilling af ordrer, når systemet beregner alt for ham.

Alternative løsningsforslag

Et alternativt løsningsforslag kunne være en udvidelse til deres nuværende webshop. Et sådant system kunne eventuelt ligge inde på selve hjemmesiden, så det både ville være nemmere, men også mere overskueligt for dem at bruge. Derved ville programmet også kunne trække salgstal ud direkte fra databasen som er forbundet med hjemmesiden.

Et andet løsningsforslag kunne også være en AI (Machine Learning), der selv kunne gå ind og vurdere ud fra deres salg af forskellige produkter, hvor mange varer der skal bestilles hjem og hvornår. Dette skulle fungere som de fleste andre AI programmer, hvor der vil være behov for at lære programmet, hvordan den skal handle ind. Denne implementering ville på længere sigt kunne finde tendenser i de forskellige sæsoner og kunne gøre indkøb mere præcis og fuldstændigt automatisk.

Scope og afgrænsninger

Scope

Programmets hovedfunktionalitet er en algoritme, som beregner, hvornår og hvor mange varer som skal bestilles hjem på et givent tidspunkt. Programmet skal også tage højde for både produktionstid og leveringstid.

Et forecast-system skal også implementeres, så systemet kan give et fornuftigt bud på, hvor meget vækst eller forfald, der vil være for en given vare. Der ønskes også en graf for varer, hvor man kan vælge en vilkårlig salgsperiode og se salgstal. Endvidere skal systemet også give en notifikation til indkøberen, om hvornår en vare skal bestilles hjem.

Afgrænsninger:

Projektet har nogle fastsatte rammer i forhold til vores eksamen, derfor skal programmet skrives i programmeringssproget C# og gøre brug af et WPF user-interface. Det giver nogle restriktioner i forhold til Powerbanken.dk's ønskescenarie; Projektet kommer ikke til at indbefatte en Mac-løsning, selvom Powerbanken.dk gør brug af Mac OS i firmaet. Derudover kommer systemet ikke til at kommunikere sammen med Powerbanken.dk's webshop og Magento, da dette ligger uden for vores pensum. Priserne på produkterne bliver heller ikke de korrekte indkøbspriser, da disse priser ikke er blevet oplyst af Powerbanken.dk.

Derudover kan vi heller ikke realisere at lave et AI, da vi ikke har nok erfaring med programmering eller maskinlæring til at kunne tage dette i brug. Dette kunne dog være en videreudvikling af programmet.

Mål og succeskriterier

Vi har i samarbejde med PO udarbejdet tre primære hovedmål, som vil give firmaet værdi nok til at implementere systemet i deres arbejdsproces. Vi har inddelt projektets mål i prioriteret rækkefølge, alt efter hvad PO vægtede højest. Det har også været vigtigt at få nedskrevet succeskriterierne, så alle er klar over, hvornår projektets mål er blevet nået.

Projektets mål	Beskrivelse	Succeskriterier
PO får præcist at vide, hvornår han skal købe varer ind.	I og med at Powerbanken vil kunne slippe for selv at skulle tænke over, hvornår de skal bestille varer hjem, vil vi optimere denne proces.	Systemet giver en notifikation, når der skal bestilles varer hjem. Powerbanken får et bedre overblik, over hvornår de er ved at løbe tør for varer.
PO kan få en estimeret vækstangivelse på alle varer på sit lager.	Det er nødvendigt for programmets funktionalitet at kunne se, hvor meget en vare enten stiger eller falder i salg.	Alle varer skal have den information vedrørende hvor mange procent den enten er steget eller faldet i forhold til den vækst, som brugeren indgiver.
PO kan tilføje og deaktivere varer.	Det ville være nødvendigt for Powerbanken.dk at kunne redigere i deres varebeholdning på sigt, da de hele tiden vil få nye varer på lager - eller slette udgåede varer.	Brugeren kan indtaste nye vareattributter i et vindue, som tilføjer dem til varelageret. Brugeren skal også kunne have muligheden for at deaktivere og aktivere samtlige varer.
PO kan se en graf af forskellige varers salgsperioder. Dette skal også virke på de brands, som de forhandler.	Powerbanken.dk ønsker at få vist en graf af varesalg over en valgfri periode.	Brugeren skal kunne vælge en til flere varer og sammenligne dem i en graf. Her vil det være muligt at vælge perioder og antal af varer, som skal sammenlignes. Det samme skal man kunne med brands.

Økonomiske gevinster

Her beskriver vi, hvordan vores system kan hjælpe vores PO til at skabe mere vækst og en højere gevinst på bundlinjen til slut på året. Disse bud er vores egne estimerede bud på, hvor meget systemet kan spare dem for på årsbasis. Da vi ikke har kunnet få adgang til Powerbanken.dk's finanser, har det været svært at vurdere priserne. Ud fra hvad vi har fået at vide af vores PO, har vi alligevel forsøgt at komme med et gæt, på hvor meget de vil kunne spare på nogle forskellige punkter.

GevinstID	Gevinstnavn	Beskrivelse	%-vis antagelse af besparelse	Gevinstejer (navn)
1	Budgetmæssig gevinst	En mere præcis varebestilling, så de kan sprede udgifterne på varerne ud bedre over hele året	Estimeret: 10%	Jacob og Christoffer
2	Socioøkonomisk leverancer	Med det nye system giver det mulighed for altid at have varer på lager, så kunderne altid får varerne inden for et par dage	Estimeret: 1-2%	Jacob og Christoffer

Ikke økonomiske gevinster

Vi har opsat nogle "bløde" værdier, som er Powerbanken.dk's ikke økonomiske gevinster. Det er gevinster, hvor det kan være svært at måle, hvor meget de præcist giver firmaet. Det er vores kvalificerede bud på, hvad der kan være gavnligt for firmaet. I skemaet nedenunder ses nogle af de punkter, vi mener systemet kan have en positiv indflydelse på.

ID	Gevinstnavn	Beskrivelse	Gevinstejer
1	Lettere arbejdsgang når der skal bestilles varer hjem	Ud over de økonomiske besparelser, vil arbejdsgangen omkring bestilling af varer også blive nemmere, efter løsningen er implementeret. De vil ikke manuelt skulle regne ud, hvornår de skal bestille en given vare hjem. De skal kun indtaste den forventede vækst i procent.	Jacob og Christoffer
2	Overblik	De får et bedre overblik over deres lagerbeholdning, og hvornår de skal bestille varer hjem	Jacob og Christoffer
3	Tid	Automatisering af arbejdsprocesserne sætter tid af til andre arbejdsopgaver	Jacob og Christoffer

Key Performance Indicators

For at kunne måle hvor godt vores system virker, har vi udarbejdet flere KPI'er, som kan bruges til at måle, om systemet virker efter hensigten, samt hvor godt det fungerer.

I vores KPI måler vi, hvor lang tid det tager for brugeren at benytte systemet, skal bestilles nye varer hjem; kontra hvor lang tid det tager for vores PO at lave samme udregning nu.

KPI	Hvor lang tid tager det for PO at bestille varer hjem?
Hvorfor måles?	Vi måler for at finde ud af, hvor meget tid vores PO kan spare, og hvor effektivt systemet er.
Hvordan måles dette?	Vi måler med et stopur for at se, hvor længe PO er om en almindelig bestilling af en vare i hånden. Derefter måler vi på, hvor hurtigt han kan gøre det i systemet.
Hvem er ansvarlig for måling?	Målingen vil blive foretaget af vores scrummaster, da han er bindeleddet mellem udviklingsteamet og PO.
Forventet målingsdato	Vi forventer at foretage målingen i starten af Sprint 3.
Forventet værdiinterval for måling	Vi forventer værdiintervallet er 90-100% hurtigere end den nuværende bestillingstid
Måling	Bestillingen af varer tager på nuværende tidspunkt en ½ - 1 time, afhængigt af hvor mange varer der er. Med systemet kan de klare hele processen på 1 ½ - 2 minutter.
Handlingsplan i fald målingen ligger udenfor forventet interval	Hvis målingen ligger uden for intervallet, er vores handlingsplan at refaktorere programmet for på den måde at nå vores ønskede interval. Derudover vil vi også kommunikere med PO for at få at vide, hvad vi har gjort galt.
Ansvarlig for handlingsplan	Handlingsplanen vil blive udarbejdet af hele gruppen i fællesskab. Dog er scrummaster ham, der skal igangsætte handlingsplanen, hvis værdiintervallet ligger uden for det ønskede resultat.

Vi har også lavet 2 andre KPI'er¹; den første måler hvor præcise vores beregninger omkring varebestilling. Den anden omhandler vores estimerede bud på, hvor meget de skal bestille hjem næste år. Disse to har vi dog ikke kunne lavet målinger på, da det kræver at systemet er i brug.

¹ Se Bilag 7 – KPI'er. Side 56

Interessenter

I forhold til vores projekt har vi fundet frem til, at der kun er en interessant, og det er vores PO. Dette skyldes, at det kun er PO, som kommer til at bruge programmet, og som kan påvirke udformningen af systemet. Vi har i gruppen snakket om hvorvidt leverandører og kunder kunne være interessenter, men da de ikke vil blive hverken direkte eller indirekte berørt af systemet eller har mulighed for at påvirke dens udviklingsproces, har vi valgt ikke at medtage dem.

Interessent	Område af projektet med særlig interesse	Holdning til projektet	Tiltag til håndtering
POs (Jacob og Christoffer)	De er interesseret i at få et system, der kan hjælpe dem med indkøb til lager, hvilket vil optimere deres arbejdsgang og eventuelt kunne spare dem penge.	De er overvejende positive overfor projektet og virker til at have mange ønskede funktionaliteter til systemet.	De lader til at have mange ideer og kan hurtigt blive meget flyvske med ideer. Derfor bør vi som udviklere sørge for, at det er de vigtigste funktionaliteter, der bliver prioriteret højest. Dette kan vi gøre ved at stille opfølgende spørgsmål til deres ideer. På den måde er vi hele tiden er sikre på, at vi har det, som er vigtigst for PO, prioriteret højest.

Handlingsplan

Vi har skrevet en handlingsplan², ud fra de tanker vi har gjort os om, hvis PO skulle efterlade os midt i processen eller stoppe al kontakt. Derudover omhandler handlingsplanen også vores gruppekontrakt samt risici. Denne handlingsplan vil forebygge tvivlsspørgsmål vedrørende eventuel sygdom, eller hvis vi på anden måde ikke kan fuldføre noget gruppearbejde.

Denne handlingsplan er dog aldrig blevet brugt, da ingen af de nævnte scenarier i handlingsplanen er blevet realiseret. Dog har den givet os nogle retningslinjer og en plan for, hvad vi bliver nødt til at gøre, hvis handlingsplanen, skulle blive relevant.

² Se Bilag 9 – Handlingsplan. Side 61

Kommunikationsplan

Vi har udarbejdet denne kommunikationsplan³, så vi altid har et udgangspunkt for, hvordan vi bedst formidler vores budskab i forhold til den part, vi henvender os til.

I dette projekt har vi haft 3 primære parter i forhold til den kommunikation, vi kommer til at have uden for vores egen gruppe. De følgende 3 er: PO, vores projektledelsesgruppe og diverse undervisere.

PO: Vedrørende kommunikationen herved ville vi altid dokumentere mødets struktur inden selve mødet. Vi stræbte derfor efter at være forberedte og professionelle på vores møder, så PO også fik noget ud af mødet, og vi fik det budskab igennem, som vi ønskede. Dette kunne f.eks. være tvivlsspørgsmål vedrørende ændringer i systemet eller mangel på data, for at forstå problemstillingen.

Derudover havde PO også det sidste ord i forhold til forløbet, da vi altid gerne ville have en accept af funktionaliteten bag det nuværende produkt.

Projektledelse: Kommunikationen med denne gruppe har været på et begrænset niveau, da vi kun havde kontakt til dem i form af enkelte møder. Dog ville vi stadig engagere os, da vi vidste, vi ville få nogle nye ideer til rapporten eller vores arbejde som helhed. Vi fik også hjælp til at blive mere reflekterende i selve arbejdsprocessen, og hvorfor vi gør, som vi gør. Det, som gjorde denne part anderledes, var, at det var dem, som skulle opsøge os, for at hjælpe os med projektet. Vi kunne derfor kun forberede os via det oplæg som de vedhæftede om mødets fokuspunkter.

Undervisere: Ved denne part kunne vi modtage mest hjælp i forhold til projektarbejdet, så allerede fra starten af besluttede vi at deltage i så mange fastlagte styringsmøder som muligt. Vi ville også hver uge planlægge møder med underviserne. Vi forbereder derefter problemstillinger, som vi i samarbejde med underviserne gennemgår. Det, vi gerne ville have ud af det her samarbejde, er følgende: At opnå en mere professionel arbejdsproces og blive bedre til at dokumentere arbejdet i gruppen.

³ Se Bilag 10 – Kommunikationsplan. Side 63

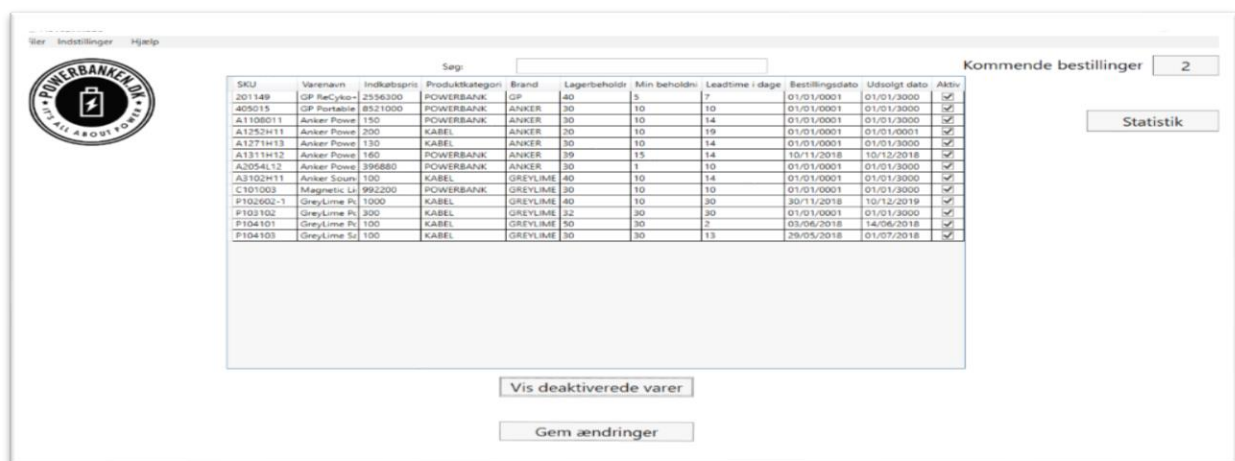
Problemformulering

Powerbanken.dk skal gennemgå en lang manuel og upræcis proces, når de bestiller varer hjem til deres lager. Dette fratager dem derfor en masse kostbar tid og penge, de i stedet kunne bruge på andre vigtigere gøremål.

Brugervejledning til systemet

Herunder ses alle de primære funktioner, systemet kan bruges til. Designet er blevet udarbejdet i samarbejde med PO, og efterfølgende testede vi også brugervenligheden med ham. Dette har gjort, at vores produkt har gennemgået flere ændringer i forhold til, hvordan programmet skal se ud. Vi har heller ikke opnået alt PO, kunne ønske sig i forhold til design. I og med det også ligger uden for vores scope, så ville vi ikke bruge længere tid på det. Dog er det noget, vi ville kunne refaktorere senere hen, hvis der bliver behov for det.

Beklageligvis har vi intet navigationsdiagram til at give et enkelt overblik over vinduerne i forhold til hinanden. Dette skyldes, at vi i starten af udviklingsprocessen ikke har haft så meget fokus på, hvordan man navigerer rundt i systemet. Vi har nærmere haft fokus på, hvilke funktionaliteter, systemet skal have. Det er først langt senere, vi er begyndt at snakke med PO omkring design af brugergrænseflade. I stedet har vi nogle screenshots, som viser systemets nuværende brugergrænseflade.

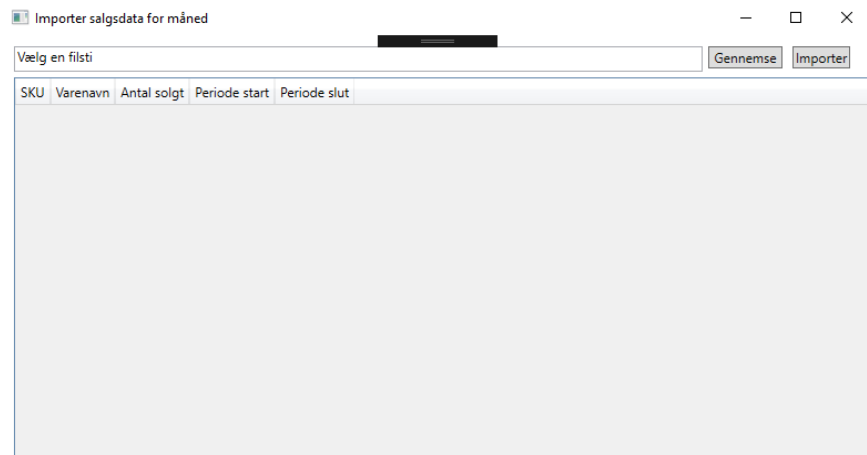


WireFrame - Startskærm

På startskærmen ses alle knapper, som åbner op for funktionalitet i systemet.

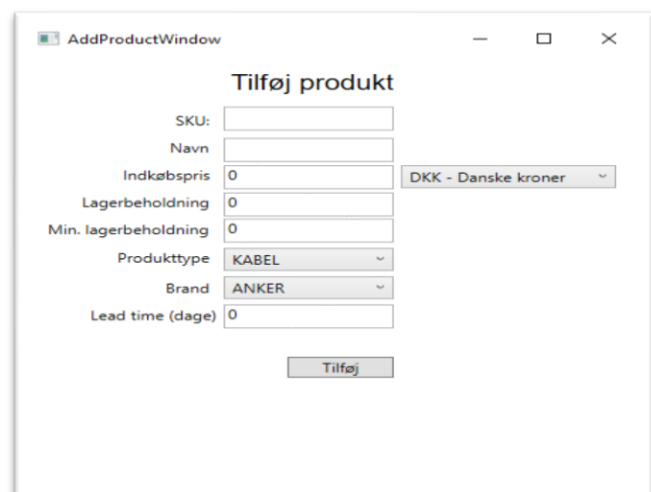
Det store "varevindue", som vises her, er en oversigt over samtlige varer både dem, som er aktive, og dem, som er inaktive. Heri kan der også rettes i beholdninger og priser.

For at Importere .csv filer skal du gå ind under knappen “Filer → “Indlæs salgsdata” på startsiden. Derefter trykker man “Gennemse”, og derefter vælger man den ønskede fil. Herefter trykker man på “Importer”, og får samtlige salgstal ind for den valgte måned. Hvis man ikke trykker “Gem i database”, bliver de ikke gemt i systemet.



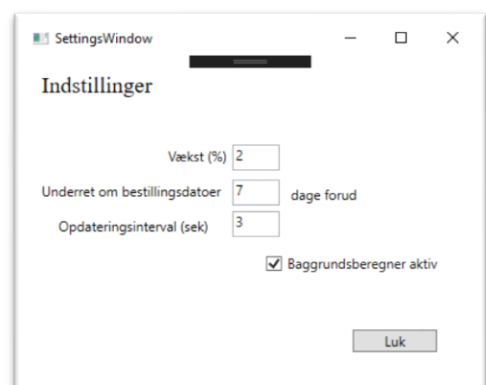
WireFrame – Import CSV

“Filer” → “Tilføj Produkt” på startsiden. Her kan man manuelt tilføje et produkt. Det er samtidigt også muligt at definere, hvilken valuta indkøbsprisen er i. Denne tilføjelse af produkterne er dog ikke uredigerbar. Du kan i visse attributter på varerne ændre deres værdi i hovedvinduet. (Se afsnit om hovedvindue)



WireFrame - Tilføjprodukt

Indstillingsvinduet giver brugeren muligheden for at indtaste, hvor mange dage før brugeren får en notifikation om, hvornår en given vare skal bestilles hjem. Hvis baggrunds-beregneren deaktiveres, får man ingen notifikationer. Det er også her, man kan indstille den forventede salgsvækst for året.



WireFrame - Indstillinger

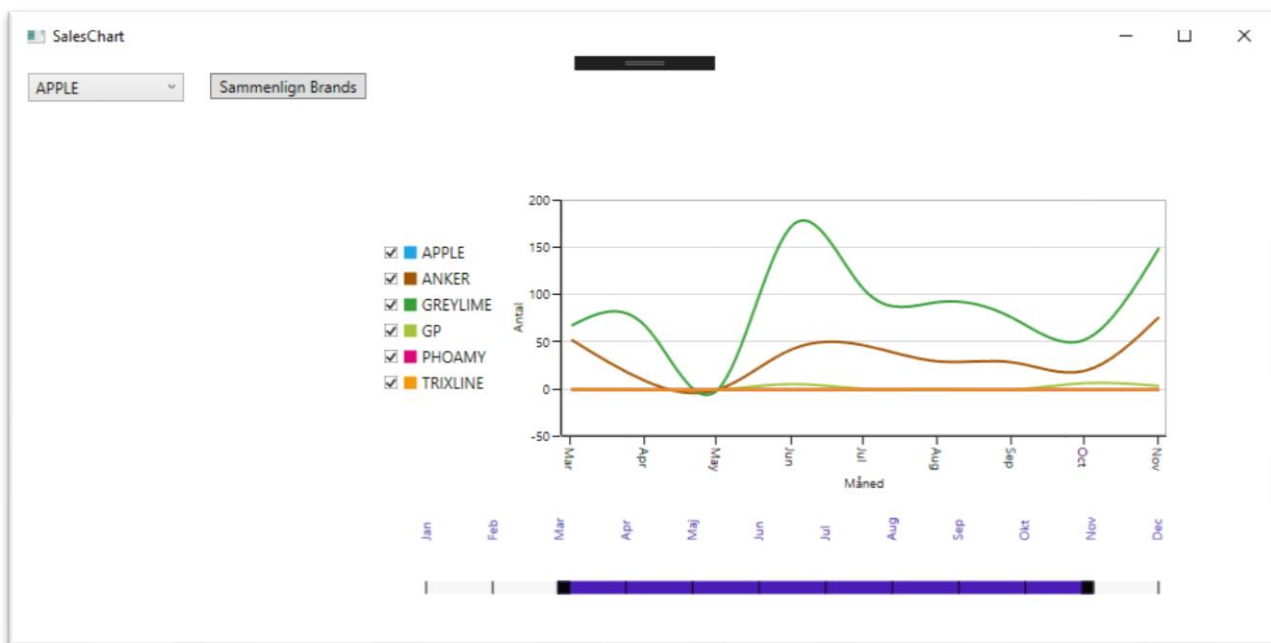
Bestillingsvinduet findes under knappen “Kommende bestillinger” på Startside. Her kan man se, hvornår brugeren skal bestille en given vare hjem. Herudover dukker der også et notifikations-vindue⁴ op på forsiden, som giver brugeren besked på, hvis varer snart overskrider bestillingsdatoen.

OrderNotificationWindow

Bestillinger de næste 30 dage

SKU	Varenavn	Bestillingsdato
P104101	GreyLime Power Stone	03/06/2018
P104103	GreyLime Sandfarvet	23/05/2018

WireFrame- Bestillingsvindue



WireFrame - Grafvindue

Grafvinduet findes under knappen “Statistik” på Startside. Her kan man få en visuel præsentation af salget for de enkelte varer samt for de forskellige brands. Det er muligt at tilkoble og afkoble varer, så man kun får vist de ønskede vare.

⁴ Se Bilag 12 – Notifikationsvindue. Side 65

Dokumentation af bestillingsdatoer

Vi har i første omgang valgt at lave vores bestillingsalgoritme så simpel som overhovedet mulig. Algoritmen kommer til at tage udgangspunkt i samtlige produkters lagerbeholdning samt deres forventede daglige salgstal for det kommende år. Ordredatoen for et produkt bestemmes ud fra den dag, hvor produktet rammer dets minimumsbeholdning. Denne minimumsbeholdning bliver prædefineret af PO, da varer har forskellige minimums beholdninger, samt hvor lang tid det tager at få produktet leveret. Denne leverancetid kalder PO for leadtime.

Når systemet skal beregne en bestillingsdato for et produkt, starter den med at tage det enkelte produkt fra en liste af produkter. Derefter samler den månedlige salgstal for produktet, hvis der eksisterer nogen, og tager fat i de salgstal, som stemmer overens med den indeværende måned. Den dividerer herefter det månedlige salgstal med antallet af dage i måneden og ganger med den vækst, som der er indberettet for hele året. (Hvis væksten eksempelvis er 20% hedder regnestykket: $\text{dagssalg} * 1.20$). Resultatet trækkes fra den nuværende lagerbeholdning, og algoritmen fortsætter med at simulere salg for de næste dage. Hvis der er månedsskrift undervejs, finder den salgstillene for den tilsvarende måned og beregner det daglige salg nok engang for den nye måned. Dette fortsætter, indtil en lagerbeholdning rammer det angivne minimum, hvorefter datoen fratrækker leadtime i dage. Datoen, som algoritmen kommer frem til, bliver så gemt i systemet. Brugeren kan derefter gå ind og se, på hvilken dato de vil ramme minimumsbeholdningen for et givent produkt. Nedenunder er den Product Backlog vi er endt med til sidst i projektet. Den er blevet udbygget undervejs efter næsten hvert møde med Jacob for at få alle hans ønsker med.

Modellering af systemet

Use-cases

Vurderingskriterier

I forhold til opbygningen af vores use-cases, har vi holdt dem til en enkelt proces. De første use-cases, der var blevet udformet, har været i formatet “brief”, da dette har været kerne-funktionalitet, som skulle være opfyldt. Dette har derfor været scenarier, såsom at “*Finde en vare*”. Herefter blev der implementeret casual use-cases, for at give indblik i både succesfulde og alternative scenarier for mere indviklede aktørmål. Disse use-cases skal desuden bestå boss-testen, så use-casen har en form for funktionalitet.

Vi vil ikke gå i dybden med fullydressed use-cases, da vi mener, det vil fylde for meget, og være uden for scopets fokus. Argumentationen for fravalget af fully dressed use-cases er at vores system er relativt småt, og vi vil ikke få noget ud af at f.eks. udforme preconditions og succesgarantier. Vi vil mene, at nogle af kriterierne for fullydressed usecases er repræsenteret i andre artefakter.

Udformning af use-cases

Vi identificerede vores to PO'er Jacob og Christoffer som primære aktører, da de to vil have direkte brugermål. De vil også være de daglige brugere af systemet. Der er ikke nogen supporting aktører i forhold til projektet. Disse kan dog identificeres ved, at de udøver en service. En mulig supporting aktør kunne være Powerbanken's webshop og deres betalingsservice Magento, men da de ikke bliver en del af vores system, vil de ikke kvalificere sig på nuværende tidspunkt som supporting aktører.

Efter vi har identificeret de primære aktører til systemet, bliver udvalgte user-stories fra product-backloggen omskrevet i hvert sprint til use-cases.

Refleksion over use-cases

Vi har i flere omgange omskrevet vores use-cases, så de ville opfylde specifikke mål for brugeren i stedet for vores umiddelbare metode, hvor vi skrev use-cases til, hvordan vi skulle udarbejde funktionalitet. Vi ville i starten gerne til at oprette kode, men efter oplæsning på emnet, stod det klart at brugeren skulle have et mål i use-casen, og den skulle opfylde boss-testen. Derfor blev use-casene både længere og opbygget med både alternative - og hovedsucces scenarier.

Herunder ses det eksempel på en use case, vi har valgt at tage med i rapporten, da den kommer ind på hvordan og hvorfor notifikationsvinduet vises. Use casen tager udgangspunkt i PBI nummer 2 i vores Product Backlog.

Use case - UserGetsNotificationToOrderProduct

Når der udregnes en bestillingsdato for et eller flere produkter, som ligger inden for den valgte periode, så vil Jacob modtage en notifikation fra systemet, som informerer ham om, at han skal bestille produktet hjem til lageret for ikke at løbe tør.

Hoved succes scenarie:

Systemet åbner et vindue, der informerer brugeren af systemet om, at en eller flere varer, burde bestilles hjem på en specifik dato for ikke at løbe tør for den vare.

Alternative scenarier:

- Indstillingerne for notifikations-vinduet har fået indsat værdier, som ikke indeholder nogle varer, som skal bestilles hjem. (F.eks. hvis notifikationen, kun skal komme frem en dag før bestillingsdagen.)
- Hvis der ikke eksisterer - eller de glemmer nogle salgsdata i databasen.
- Hvis systemet opdager en fejl i den eksterne database
-

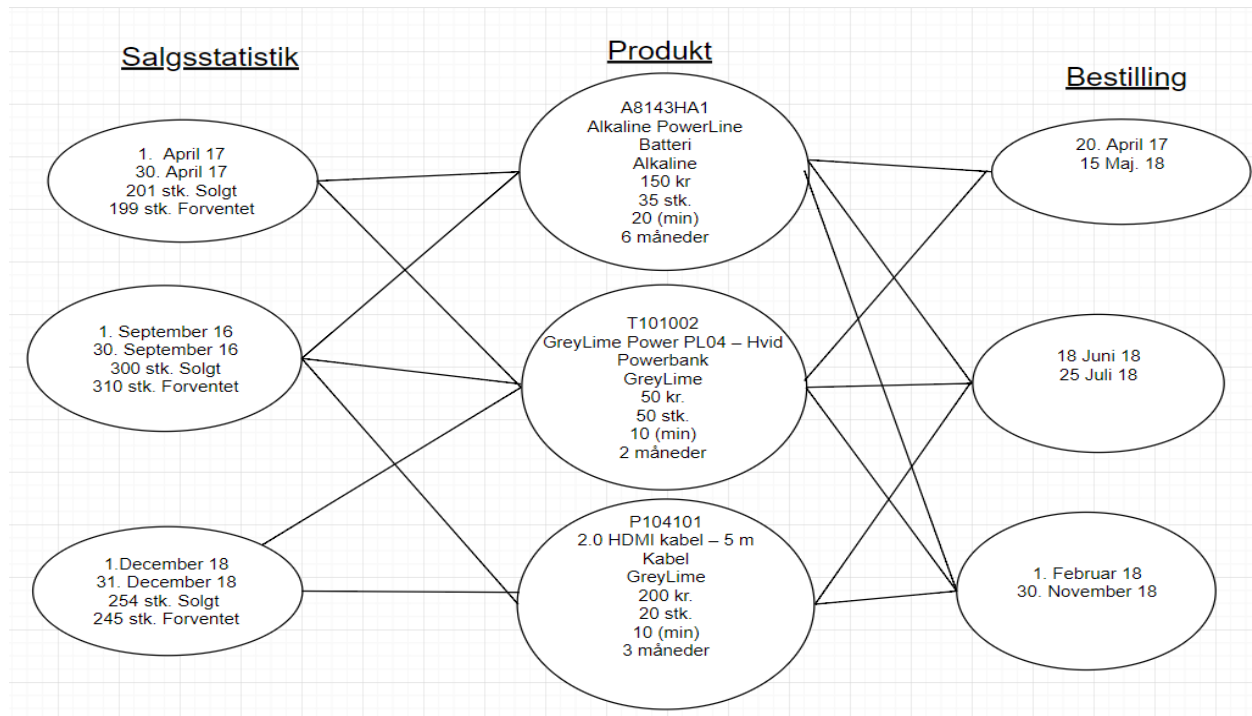
Objektmodel

For at lave vores objektmodel, har vi har taget navneordene i vores PBI'er og dertilhørende user stories. Ud fra disse har vi diskuteret, hvilke der bør være objekter, og hvilke der skal være attributter under de forskellige objekter. Denne model har også været refaktorert flere gange, da udarbejdelsen af objektmodellen har ændret sig næsten fra gang til gang, vi har været i kontakt med PO. Vi har også lavet forbindelser mellem objekterne og hvordan de er forbundet. Dette har været vigtigt at vise, så vi ved hvordan domænemodellen kommer til at udforme sig.

Opbygning af objektmodellen

Vi er kommet frem til, at et produkt indeholder en SKU, et navn, en type, en indkøbspris, en mængde, en minimumsmængde og en leadtime. Et produkt-objekt er desuden forbundet til flere salgsstatistik-objekt, som indeholder en start- og slutdato, en solgt mængde og et forventet salgstal.

Derudover kan et produkt have flere bestillingsobjekter, som hver især indeholder en bestillingsdato og en udsolgt-dato.



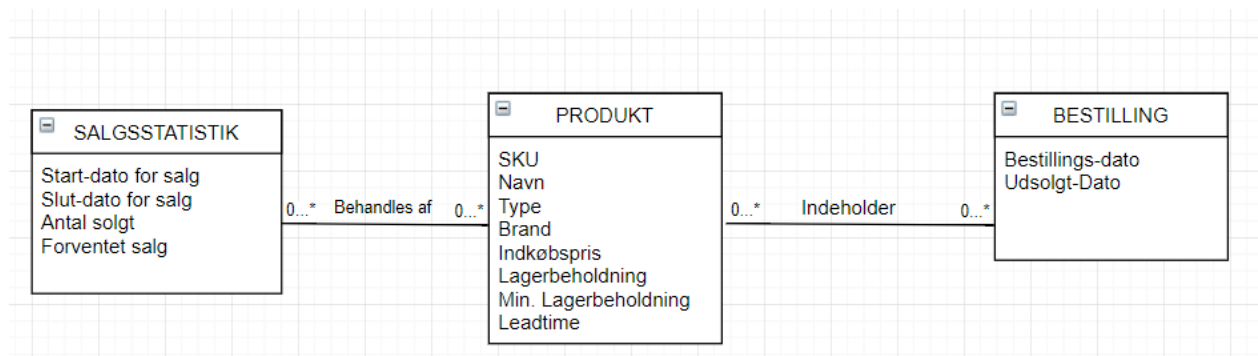
Objektmodel

Domænemodel

Vurderingskriterier:

Vores strategi for at finde konceptuelle klasser og opbygningen af domænemodellen har taget udgangspunkt i vores objektmodels objekter og dens kardinaliteter.

Vi har derudover taget vores use-cases og fundet navneord som kunne indikere mulige klasser i vores domænemodel for Powerbanken.dk. Endvidere har vi også sorteret de konceptuelle klasser fra, som ikke er en del af vores scope, såsom kunde- og leveringsklasser, men som dog stadig eksisterer i Powerbanken.dk.



Domænemodel

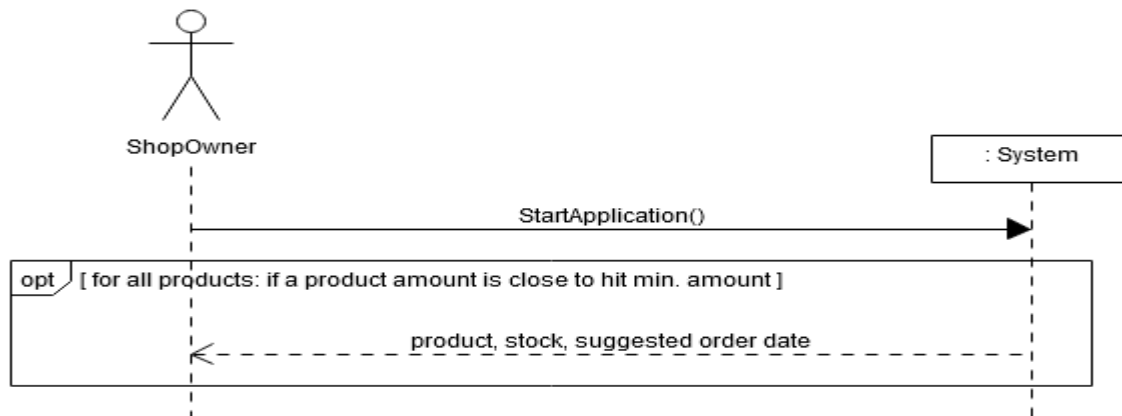
Refleksion

Vi havde allerede fra starten en god fornemmelse af, at vores to modeller skulle bygges op ud fra en konceptuel klasse kaldet “Produkt”, og en “Bestillings”-klasse. Dog mente vi oprindeligt, at bestillings-klassen skulle være en kalender i stedet. Dette kom vi dog hurtigt fra, da dette mere var en visuel repræsentation i systemet. Derudover havde vi også en leadtime-klasse (produktionstid + leveringstid), som vi kunne bruge i forhold til databasen, men efter vi fik det vendt med hinanden, blev vi klar over at leveringstid og produktionstid, mere var attributter i Produkt-klassen, så vi ville ikke have Leadtime som en individuel klasse alligevel.

SSD - UserGetsNotificationToOrderProduct

Når vi har udarbejdet SSD’er i vores projekt, har vi opstillet vurderingskriterier, der tager udgangspunkt i Larman-bogen. For at finde frem til de forskellige interaktioner har vi brugt vores use cases.

På systemsekvensdiagrammet nedenfor ser vi butiksejeren, som starter systemet. Straks efter kører systemet beregningen af ordredatoer i baggrunden og tjekker, om den skal give brugeren en notifikation. Såfremt der er et eller flere produkter, hvis ordredato ligger inden for den valgte periode (som standard 7 dage), returnerer systemet en underretning til brugeren, som vil kunne se produktnavn, SKU, ordredato, nuværende lagerstatus. I fald der ikke er nogen produkter, som skal bestilles hjem inden for den valgte periode, returneres der intet. Af denne grund ligger pilen, som går fra systemet til brugeren, inde i en interaktion frame med nøgleordet “opt”.



SSD - UserGetsNotificationToOrderProduct

SOC - UserGetsNotificationToOrderProduct

Vores udarbejdede SOC'er skal fungere som en slags "kontrakt" for interaktioner med systemet, som udarbejdes ud fra en interaktion i en SSD. De skal derfor vise, hvilke ændringer der bliver foretaget i systemet, når brugeren interagerer med systemet. Når vi har lavet SOC'er, har vi hentet kvalitetskriterier fra Larman-bogen.

Systemoperationskontrakten, som ses nedenfor, viser først, hvilken tilstand systemet er i, før der beregnes ordredatoer for produkter, og en notifikation kan sendes til brugere. Der nemlig nogle betingelser, som skal være opfyldt: der skal være angivet en vækst for året, baggrundsberegneren skal være aktiv. Efter operationen er kørt, er der ingen post-conditions, eftersom systemets tilstand ikke har ændret sig. Dvs. at der er intet slettet, oprettet eller ændret.

Operation: GetOrderDatesForProducts(Orderdate, ProductInfo)

Cross reference: *UserGetsNotificationToOrderProduct*

Pre-conditions:

- The background calculator has not been deactivated by the user.
- User has typed a number in percent under "Growth" in the "Settings-window".

Post-Conditions:

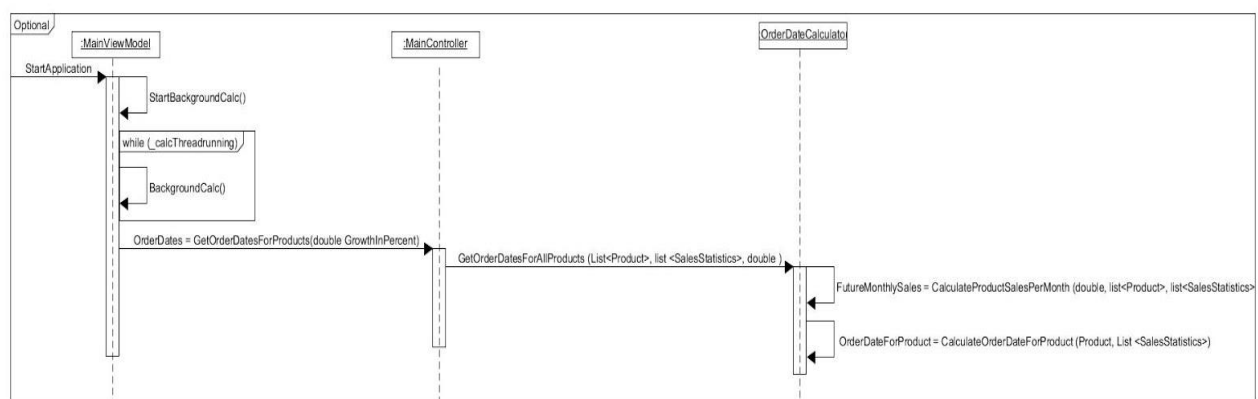
- No actions.

SD – UserGetsNotificationToOrderProduct

Refleksion af SD

Når vi har udarbejdet vores SD'er, har vi taget udgangspunkt i Larmans UML. Vi vil derfor have en klar rød tråd igennem vores artefakter, og det ses også i vores SD. Den tilknyttede SSD's sekvens, vil være synlig i starten af SD'en. Derudover har vi også haft fokus på at lave SD'erne, før vi koder, så alle har været klar over, hvordan de forskellige objekter interagerer med hinanden. Dette har gjort det nemmere at overskue funktionaliteten, da vi allerede har fundet sigende navne til vores metoder og parametre.

Den tilhørende SD, vi har valgt at vise i rapporten, har vi taget med, fordi den faktisk ikke beskriver en brugerinteraktion med systemet. Denne funktion sker kun, hvis en vare på lageret er ved at være udsolgt, så bliver metoden aktiveret. Den sender derfor en notifikation til brugeren om, hvornår brugeren skal bestille varen hjem, så han ikke får denne vare udsolgt. Dette er også grunden til, at systemet aldrig returnerer et output tilbage til brugeren. Det er implicit vist ved "OrderDates = GetOrderDatesForProducts(double GrowthInPercent)" i SD'en. Her vises, hvad brugeren modtager, når systemet opdager en vare, som er ved at være udsolgt. Måden, hvorpå vi kører alle varer igennem for at se lagerbeholdningen, er, at vi gør brug af tråde. StartBackgroundCalc og BackgroundCalc, starter hhv.tråden, og BackgroundCalc får udregningen fra de andre metoder i systemet.



SD - UserGetsNotificationToOrderProduct

Kode - UserGetsNotificationToOrderProduct

Nedenfor ses en af de metoder, som vi bruger til at beregne ordredatoerne for alle produkter. Vi har ikke kunne inkludere al kode, som er relevant for operationen, i rapporten, men resten er at finde som bilag, se i vores solution-fil. Eksemplet nedenfor er dog den mest væsentlige del. Hvordan, koden fungerer, står beskrevet i det tidligere afsnit, "Algoritme til beregning af bestillingsdatoer".

```
7 references | Daniel Stuhr Petersen, 8 days ago | 1 author, 1 change
public DateTime OrderDate { get; private set; }
1 reference | Daniel Stuhr Petersen, 8 days ago | 1 author, 1 change
public DateTime SoldOutDate { get; private set; }
2 references | Daniel Stuhr Petersen, 8 days ago | 2 authors, 2 changes
public void CalculateOrderDateForProduct(Product product, List<SalesStatistics> futureSalesForProduct)
{
    const int YEAR_LIMIT = 3000;
    DateTime currentDate = DateTime.Today;
    int dailySale = GetDailySaleForMonth(currentDate, futureSalesForProduct);
    Product productCopy = new Product(product.Name, product.SKU, product.PurchasePrice, product.StockAmount, product.MinStock, product.Type,
        product.Brand, product.LeadTimeDays, product.IsActive);
    while (productCopy.StockAmount > 0) //simulerer salg.
    {
        if (!currentDate.Month.Equals(currentDate.AddDays(-1).Month)) // hvis måneden er skiftet.
        {
            dailySale = GetDailySaleForMonth(currentDate, futureSalesForProduct);
        }
        productCopy.StockAmount -= dailySale; // fratrækker det daglige antal salg fra lagerbeholdningen.

        if (productCopy.StockAmount == productCopy.MinStock)
        {
            this.OrderDate = currentDate.AddDays(-product.LeadTimeDays);
            if (this.OrderDate < DateTime.Now) this.OrderDate = DateTime.Now;
        }
        if (currentDate.Year < YEAR_LIMIT) currentDate = currentDate.AddDays(1); // vi tæller frem med én dag, hvis ikke tidsgrænsen er nået.
        else break;
    }
    this.SoldOutDate = currentDate;
}
```

Arbejdsprocessen af kildekoden

For at sikre vores kodekvalitet har vi kigget på, og benyttet os af Grasp og SOLID principperne, samt brugt MVVM. Vi forsøger så vidt som muligt at opfylde alle kravene fra disse principper, der hvor vores evner rækker til, men primært der, hvor det giver mening. Vi har prøvet at opretholde en høj samhørighed og lav kobling i vores system. Et af de steder dette er tydeligst er i vores controller-lag over vores DataAccess- og Model-lag. Denne klasse bliver brugt som et mellemlid mellem vores domænelag og vores ViewModel-lag. Dette viser, at vi har gjort os nogle overvejelser vedrørende lav kobling. Vi stræber meget efter, at en metode i en klasse kun skal have et formål og gøre en ting. Derfor var det en god ide at benytte et controller lag, så vi ikke får meget høj kobling. Navngivning er også noget, vi har lagt meget vægt på i vores program. Det er vigtigt for os, at vi kan gå tilbage til det, vi tidligere har lavet, og forstå, hvad det helt præcist gør. Derfor er det vigtigt at navngivningen er konkret, og at metoderne gør det, som de er blevet navngivet til at gøre.

Samtidig har det også været en stor hjælp at være i stand til at se, alene ud fra navngivning, hvad de enkelte attributter er - såsom backing-variabler har en understregning foran, og at den dertilhørende property starter med stort.

Vi har i tidligere projekter ikke været så gode til at overholde princippet om Open/closed. Det har vi i denne omgang valgt at lægge mere vægt på, og det er dermed også forbedret en hel del. Klasser burde som udgangspunkt ikke have tilgang til metoder, som de ikke skal bruge. Dette prøver vi derfor at håndtere efter bedst mulige evne.

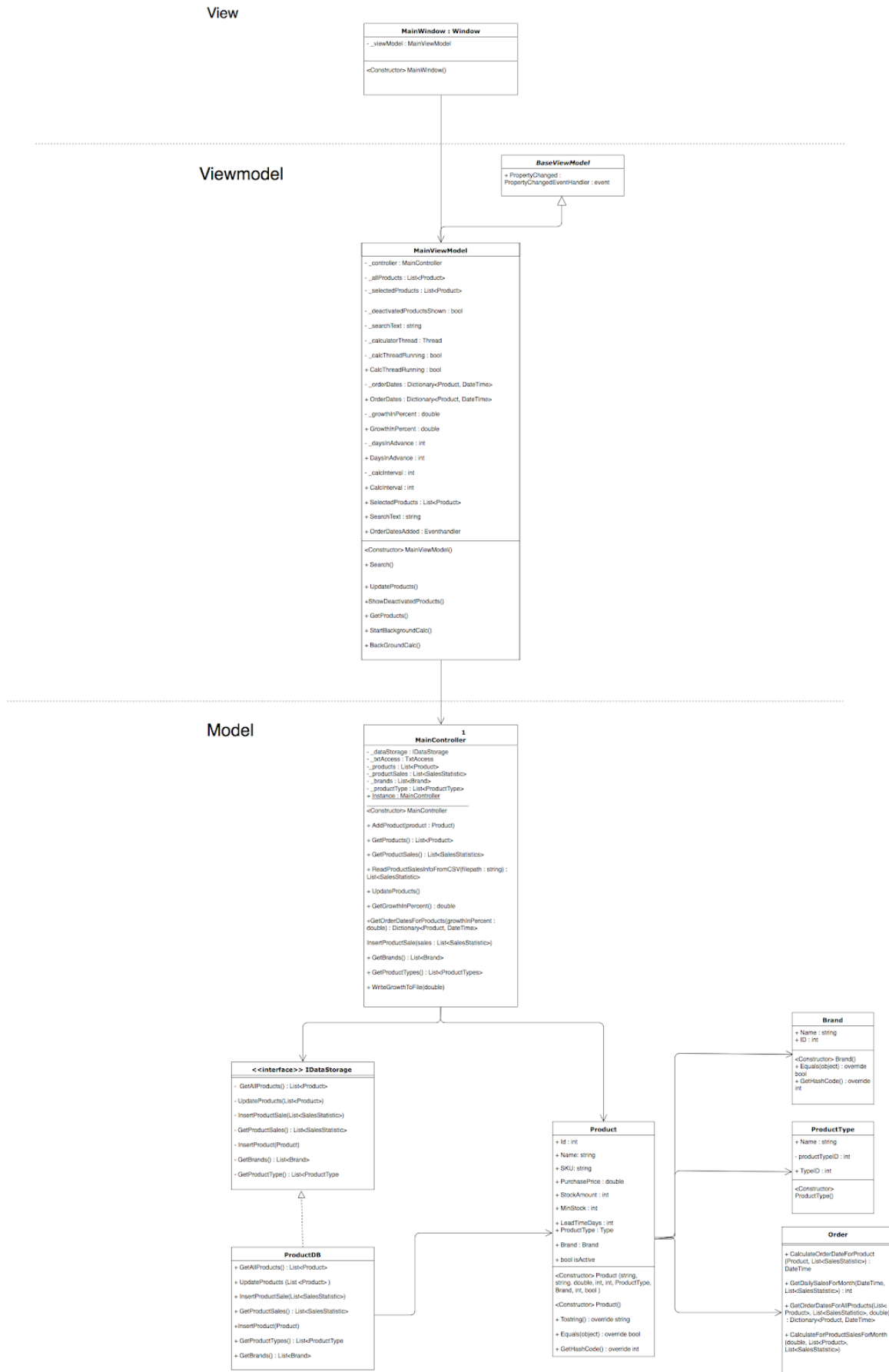
Design Class Diagram

I vores DCD finder vi alle de konceptuelle klasser, som vi allerede har identificeret i domænemodellen. Dette ses med klasser såsom Product, Order og Statistic. Derudover ses i attributterne i domænemodellens klasser, men også i software-klasserne som properties.

Når vi vurderer kvaliteten af vores DCD, har vi lagt fokus på lagdeling, overskuelighed og navngivning. Vi har derfor lagdelt DCD i forhold til vores MVVM, så man kan få et hurtigt overblik over de forskellige klasser, og hvordan klasserne er forbundet ned gennem alle lagene. Herudover viser vi visuelt alle properties, variabler og metoder med deres respektive access modifiers, for at gøre det klart hvilke ting, man kan tilgå forskellige steder i systemet.

Vi har i vores view-klasser fravalgt at anvise samtlige event-listeners (button-click etc.), comboBoxes mm. Dette er blevet gjort med henblik på overskueligheden i vores DCD, da vi gerne vil have den som en oversigt over systemet, og hvordan de forskellige softwareklasser hænger sammen, og hvilke der har kendskab til hvilke.

Design klassediagrammet, som ses herunder, viser de relevante klasser for beregning af ordredatoer. Det er tværsnit gennem de forskellige lag, view, viewmodel og model. Nogle af klasserne vil man formentlig være bekendt med efter at have set vore domænemodel. De andre softwareklasser er nogle, som har været vist sig nødvendige at tilføje. Fx. har vi været nødt til at tilføje Brand som en klasse, og ikke en attribut, som domænemodellen viser. Dette har vist sig nødvendigt i forbindelse med for eksempel graferne over salgstal, og når man skal tilføje et produkt.



Valg af arkitektur

MVVM

Vi skal udarbejde en wpf-applikation, og ved at bruge MVVM har vi mulighed for at afkoble komponenterne (View, Viewmodel og Model) fra hinanden, hvilket gør det muligt at bytte komponenter ud nemt og hurtigt.

Vores PO, har som sagt lagt stor vægt på en Mac-implementation, derfor mente vi, det var mest optimalt at have al logik, som var med visning at gøre, i viewmodel laget, og det eneste, vi derefter skulle lave om, var vores view. Dog ligger mac-implementationen efter projektets afslutning, hvis PO ønsker at vi arbejder videre med systemet. Foreløbigt bliver det kun til en Windows-app, eftersom det er, hvad WPF tillader os at lave.

Vi vil heller ikke gøre meget ud af den visuelle brugergrænseflade, og sætte os ind i f.eks. Pages frem for vinduer og lære denne funktionalitet. Vi bliver heller ikke krediteret på vores GUI, så vi ville hellere fokusere på funktionalitet i stedet for design.

PO ønsker dog rigtig meget en statistik-funktion, der grafisk kan vise ham vækst og forfald både for forskellige brands, men også for flere produkter. Argumentationen for at lave dette er, at det giver stor værdi og brugbarhed til PO. Dette vil også videre blive diskuteret i afsnittet: *“Grafik Visualisation”*⁵.

Domænelag

I vores domænelag har vi valgt at have et controller-lag mellem vores data-access og domænelag. Dette blev gjort fordi, vi i starten havde brug for at dataaccess skulle kende til domænelaget, og domænelaget havde brug for objekter fra data-access. Controllerlaget blev derefter lagt ind, så vi fik en lav kobling mellem de to, og give os muligheden for at delegere arbejdsvejen mellem domænet og data-access ud til controllerlaget.

Viewmodel

Vi ville have et 1:1 klasseforhold mellem viewmodel og view. Dog er der lige to enkelte undtagelser, da vi har klassen “BaseViewModel”, som der implementerer INotifyPropertyChanged, som vi bruger, når vi databinder på forskellige properties og objekter. Derudover gør vi også brug af interfacet “ICommandHandler”, så vi kan gøre brug af ICommand. Vores logik har vi som sagt samlet i viewmodel-laget, så vi ikke får en høj kobling til hverken domæne- eller viewlaget.

⁵ Se Afsnit “Grafisk Visualisation”. Side 37

View

Vi har ikke opfyldt MVVM's struktur korrekt i vores view. Vi har nemlig haft nogle problemstillinger, da vi blev nødt til at implementere noget logik i viewet, dette har medført flere diskussioner om, hvordan vi skulle undgå netop dette. Vi kom derfor frem til, at vi kunne bruge ICommand, men da vores erfaring begrænsede os, droppede vi det, da det resulterede i at alt for meget så skulle refaktoreres. Derfor, prøvede vi at lave det for en enkelt klasse som et forsøg på læring og som en lille ekstra udfordring. Dette er blevet gjort i klassen "AddProductViewModel", så der er dokumentation for vores forsøg på en ICommand implementation. Argumentationen for at vi ikke har valgt at ændre alle kald fra view og ned i vores viewmodel, er baseret på, vi gør brug af mange message-boxes. Disse er komplicerede at bruge, når vi har med ICommands at gøre. Det vil nemlig tvinge os til at give vores viewmodels ansvar for at vise messageboxes ved errors, bekræftelser osv. for brugeren. Vi mener ikke at dette ansvar bør tilfalde en viewmodel, men derimod viewet. Der er naturligvis også den løsning, at vi commandbinder til viewmodel, og viewmodel trigger et event ved systemfejl og bekræftelser, som view'et lytter på, så det ved, når det skal vise messageboxes. Vi mente dog ikke, at vi burde gå så meget op i dette, eftersom den løsning, som vi har med en lille smule kode i codebehind-filen, er overskuelig og giver mening.

```
public ICommand AddProduct
{
    get
    {
        return new CommandHandler(() => AddProducts(), true);
    }
}

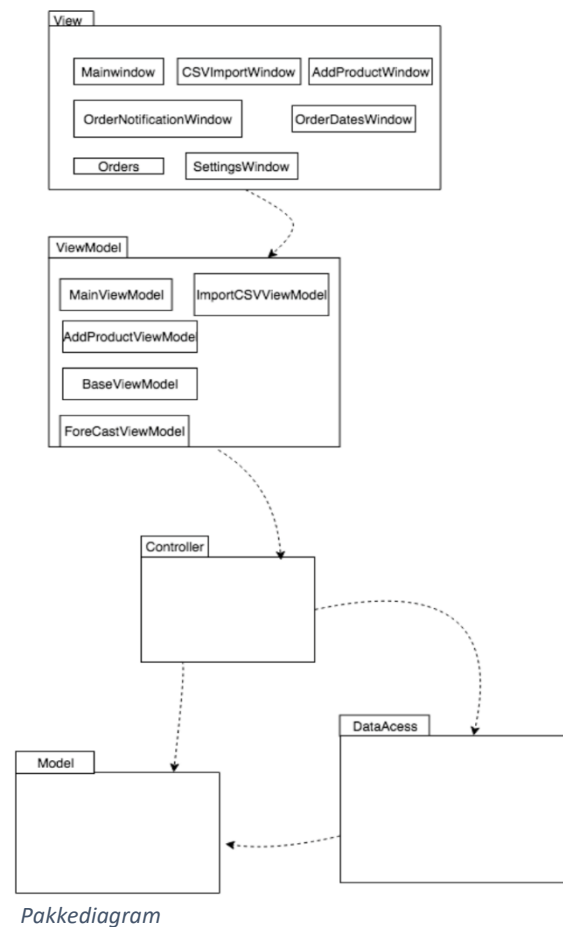
public void AddProducts()
{
    try
    {
        if (Name != string.Empty && SKU != string.Empty && Type != null && Brand != null)
        {
            Product prod = new Product(Name, SKU, PurchasePriceDKK, StockAmount, MinStock, Type, Brand,
            LeadTimeDays, IsActive);
            controller.AddProduct(prod);
            ProductAdded.Invoke(Name, null);
        }
        else
        {
            throw new Exception("Produktet blev ikke gemt - tjek din indstatning");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Lagdeling

Nedenunder ses vores nuværende pakkediagram.

Med hvert lag inddelt i mapper, med de forskellige klasser i. De stiplede pile der går fra “mappe” til mappe viser deres afhængighed.

Pakkediagrammet afslører, at vi har valgt at medtage en controller, selvom denne ikke er en del Model-View-ViewModel-arkitekturen. Dette har vi gjort for at lave en lavere kobling mellem vores viewmodel-klasser og vores dataaccess-klasser, da koblingen ellers vil være ret stor herimellem. Vi har en viewmodel for hvert view, og mange af dem har brug for data, som ligger nede i DataAccess. Vores controller sørger for at kalde fra metoder i de to underliggende lag og fungerer samtidigt som et repository for produkter, der indlæses fra databasen.



Pakkediagram

Testing

Vi vil rigtig gerne bruge arbejdsmetoden TDD. Dette

er blevet fastsat, så vi også har en plan for kvalitetssikring af den kode, som vi udleverer til Powerbanken.dk. Derudover vil vi også have noget at påvise, hvis der bliver fremsat tvivlsspørgsmål vedrørende funktionaliteten i programmet.

Vi har dog undervejs i første sprint gjort os nogle overvejelser om hvordan vi tester, og hvad vores kriterier for testing egentligt er, og har fundet ud af, at vi egentligt ikke opfylder TDD. Denne arbejdsmetode har vi genovervejet, og i de sidste sprints arbejdede vi mere metodisk med TDD. Ment at vi vil opstille vores tests først og derefter skrive kode, der kun lige kan bestå, de tests vi stiller op, for derefter at udbygge testen. Det samme bliver herefter gjort med koden, indtil vi til sidst står med den ønskede funktionalitet. Dette har givet os et bedre indblik i måden, hvorpå man arbejder på en sikker og professionel måde.

Databaseopsætning

Valg af database:

Da vores projekts størrelse ikke bliver meget større end et lagersystem med en ordre beregning i, kan man argumentere for, vi bare kunne gemme information i en streamreader-fil. Hvis det senere blev ønsket at opskalere og refaktorere løsningen, vil det gøre det uoverskueligt og svært at redigere det i en streamreader-fil. Derudover er tekstfiler meget svære at opdatere med data, så det havde ikke været en hensigtsmæssig løsning.

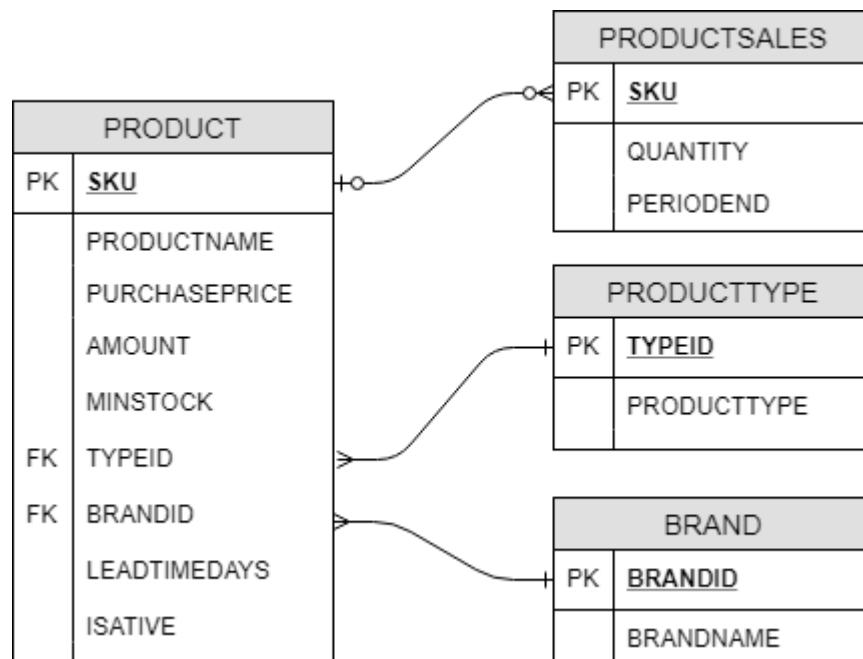
Derfor faldt valget på MSSQL, som var det oplagte valg grundet, at vi tidligere har arbejdet med et projekt, der har gjort brug af MSSQL Server. MSSQL giver os mulighed for at gemme store mængder data i en database, samt større fleksibilitet, hvis der sker ændringer i tabellerne. Da lagersystemet for Powerbanken.dk indeholder hundrede-vis af varer, og MSSQL originalt er lavet til at servicere store firmaer, har det virket som et udmærket redskab, som alle i gruppen har kendskab til. De er også flere ansatte i virksomheden, som skal have adgang til systemet, og dette gøres nemmest med en database.

Normalformer

Når vi har sat vores tabeller op i databasen, har vi haft de fire første normalformer for øje, første til tredje normalform og Boyce Codd's normalform. Nedenfor ses de fire tabeller i databasen, hvori systemet persisterer data.

Vi har opfyldt første normalform, fordi der ikke forekommer flere attributter med ens værdier i nogle af tabellerne. Desuden eksisterer der en unik primærnøgle i hver tabel. I Product-tabellen er det SKU, som er primærnøgle. Dette er den mest praktiske, da hver vare har en unik SKU. Det er desuden også SKU'en som PowerBanken.dk i forvejen bruger til at identificere de forskellige varer i deres webshop og på deres lager. Anden normalform er også opfyldt, fordi alle attributter de enkelte tabeller er afhængige af hele primærnøglen, hvad enten der er tale om en sammensat nøgle eller ej. Der er ingen attributter, som kan determineres alene ud fra en del af primærnøglen.

Derudover opfylder vi også tredje normalform, da vi ikke har nogen transitive afhængigheder i nogen af vores relations. Der er ikke nogen ikke-nøgle-attributter, som kan determinere andre attributter i en tuple. Dette skulle gerne kunne ses i samtlige tabeller. Boyce Codd's normalform er også opfyldt, eftersom alle vores primærnøgler, determinerer alle attributter i deres respektive tables, og ingen attributter determineres af andet end primærnøglen.



ER diagram

Versionsstyring

Vi har i vores projekt brugt Github. I Github har vi oprettet et fælles projekt “EksamensProjekt”, hvor vi holder alle vores filer. Det inkluderer, udover vores Visual Studio projekt, også projektlog, artefakter og Word-filer vi alle skal have adgang til. Vi har i vores projekt oprettet branches i vores Github projekt, hver gang vi har påbegyndt en større ændring/tilføjelse til projektet. Det kan f.eks. Være tilføjelsen af SyncFusion, som er en tilføjelse til Visual Studio, der tillader os at implementere grafer. Branches har derfor været en oplagt funktion at vælge, da vi derved har en “gammel” version uden SyncFusion, som vi altid har kunne gå tilbage til, skulle SyncFusion være overflødig eller ikke virke korrekt.

Objektorientert programmering

Polymorfi

Vi har i vores system benyttet polymorfi, når vi har lavet den klasse, som skal læse og skrive data til det persisterede lager. Selvom vi ret tidligt blev enige om at bruge MSSQL og stored procedures til at persistere data, ville vi gerne have, at dette skulle kunne udskiftes med et andet DBMS. Det kunne nemlig vise sig, at det webhotel, som driver Powerbankens webshop bruger MySQL,

PostgreSQL eller noget helt tredje. Vi har derfor lavet interfacet “IDataStorage”, som implementeres af vores nuværende databasefacade. Dette interface gør det let at vedligeholde flere databasefacader, eftersom man har en kontrakt på, hvilke metoder de skal indeholde. Man kommer altså ikke til at glemme at vedligeholde nogle af dem ved en fejl.

Vi har også gjort brug af polymorfi i testklassen “TestDB”, som implementerer det før omtalte interface. Dette har vi gjort fordi, vi ville have en mock af databasen, så vi kan både slette og ændre varer i den rigtige database, uden dette skal påvirke nogen af vores tests. Dette har også været med henblik på, at vi kunne udforme vores database, og hvordan vi ville have den til at se ud.

Vi har dog ikke brugt polymorfi i forbindelse med vores domæneklasser, fordi det ganske enkelt ikke har vist sig nødvendigt. Fx. har det ikke været nødvendigt at bruge en abstrakt klasse til produkter, eftersom alle produkter kommer til at indeholde de samme attributter (pris, type etc.) og de samme metoder. I fremtiden vil man i stedet kunne implementere dette, såfremt det viser sig nødvendigt, men på nuværende tidspunkt vælger vi helt bevidst at lade være.

Brug af Threads

Et af PO's ønsker er at få en underretning, når det er ved at være tid til at bestille en given vare hjem. For at systemet kan imødekomme dette, mens den står parat til at udføre andre opgaver sideløbende, har det været nødvendigt at gøre brug af tråde.

Hvis systemet skulle køre sekventielt, ville PO skulle vente med jævne mellemrum, når der beregnes bestillingsdatoer for samtlige produkter. Dette ville der gå meget lang tid med, og det ville være spildtid, hvis det viste sig, at ingen af de beregnede bestillingsdatoer var inden for nærmeste fremtid. Af denne grund har vi valgt at implementere en tråd, som kan udføre disse beregninger i baggrunden og underrette PO, hvis en af bestillingsdatoerne er tæt på dags dato. Tråden kører beregningen for samtlige aktive produkter i sortimentet med et fast tidsinterval, som kan ændres fra menupunktet “Indstillinger”.

Fra hovedvinduet har man adgang til en liste af produkter og deres foreslåede bestillingsdatoer, som opdateres løbende af baggrundstråden, så længe applikationen kører. Hvis beregningen lige har kørt, og der er blevet tilføjet en eller flere til listen, får brugeren en notifikation med info om, hvor mange varer med tilhørende bestillingsdatoer, som er blevet tilføjet.

Tråden initieres det øjeblik, at systemet startes. Selve starten sker i MainViewModel, hvori der står defineret en metode, som kaldes i konstruktøren. Den startede tråd kalder en anden metode, som kører i en while-løkke, og som kalder ned gennem controlleren og videre til OrderDateCalculatoren. Dette kald sker med et fast interval. Til at sikre dette interval, er der brugt en Thread.Sleep(). Vi har desuden trådsikret vores system, ellers risikerer vi, at systemet bryder sammen, når man tilføjer produkter. Betingelsen for at trådens while-løkke kører er en bool, som ligger i samme klasse som instansvariabel. Fra UI har brugeren mulighed for at ændre denne bool-værdi og herved terminere tråden og starte den igen efter ønske. De har ligeledes mulighed for at ændre intervallet for baggrundsregningerne. Nedenfor ses den metode, som ved systemets opstart initierer tråden og den metode, som tråden kører.

```
private void StartBackgroundCalc()
{
    _calculatorThread = new Thread(BackgroundCalc);
    _calculatorThread.Start();
}

private void BackgroundCalc()
{
    while (_calcThreadRunning)
    {
        ProductNotifications = _controller.GetOrderDatesForProducts(GrowthInPercent);
        Thread.Sleep(CalcInterval * 1000);
    }
}
```

Teknikken bag tråde

Grunden, til at vi har valgt at køre vores beregning af ordredatoer i en separat tråd, er at man udnytter den tid, hvor systemet venter på input fra brugeren. Hvis vi valgte at køre i én tråd, ville systemet stå og lave ingenting, indtil den modtager input fra brugeren. Vi har gjort det muligt, at systemet kan udnytte CPU'en til at lave beregninger, mens brugeren navigerer rundt i systemet og laver andre ting.

Programmets tråde kan desuden, på flerkernede computere som understøtter multitasking, få køretid samtidigt. Under alle omstændigheder vil systemet aldrig være blokeret pga. manglende input. Dette er en fordel, i og med at meningen med systemet i fremtiden er, at det skal kunne hive salgsdata ud og lave beregninger, mens computeren ikke bliver brugt, eventuelt natten over, eller

når det er mest hensigtsmæssigt. Derfor har vi valgt at allokere nogle tråde til beregning, så dette kan ske i baggrunden konstant, uden at bruge alt for meget processorkraft.

Valutaomregner

PO ønske at når man tilføjer et produkt så skal kunne skrive produktets indkøbspris i en anden valuta end DKK. Herefter skal systemet selv kunne omregne prisen til DKK, når der klikkes ”Tilføj produkt”, så prisen gemmes i databasen som danske kroner. Kursen, som prisen skal udregnes efter, skal desuden opdateres dagligt, da den ellers hurtigt vil blive forældet.

For at imødekomme PO’s krav søgte vi rundt omkring på nettet efter et API, som vi kunne kalde én gang dagligt for at hente valutakurser. Vi fandt dog hurtigt ud af, at de fleste kostede penge, og at de, som var gratis, ikke kunne levere det, som vi skulle bruge. Derfor måtte vi forsøge os med nogle alternativer. Løsningen endte med at blive en implementering af en web-klient i C#, der ved systemets start henter en HTML-fil (<http://borsen.dk/kurser/valuta/dkk.html>), som indeholder en liste med opdaterede valutakurser. HTML-filen bliver så gennemlæst af en StreamReader, som vi har fået til at kigge efter nogle kendetegn i HTML-koden. På den måde kan vi trække valutakurserne ud af filen og være sikre på, at valutakurserne, som bruges i vores system, er up to date.

Allerede før vi gik i gang med at implementere denne løsning, var vi bevidste om, at den måske ikke er helt langtidsholdbar. På nuværende tidspunkt er systemet afhængigt af Børsens webdesign for at kunne hente valutakurserne. Dvs. vælger Børsen at lave større ændringer i deres hjemmesides layout, risikerer vi, at vores system ikke længere kan hente valutakurserne. Brugeren vil i givet fald få en fejlmeddelelse ved systemopstart, og det vil blive nødvendigt lave meget af den logik, som er forbundet med StreamReaderen, på ny. Vi har dog alligevel valgt denne løsning, da den fungerer nu og her, når vi skal fremvise systemet. Hvis systemet bliver taget i brug, vil vi nok anbefale vores PO at afse midler til at betale for en API tjeneste, som er mere holdbar og kræver mindre vedligeholdelse i længden

Grafisk visualisation

Vi har i sprint 2 undersøgt, hvordan man laver en grafisk præsentation af Powerbanken.dk's salgsdata. PO har nemlig ønsket en graf, som viser tidligere såvel som fremtidige månedlige salgstal for samtlige produkter filtreret på brand- og produktniveau.

Vores løsning er indtil videre at bruge Syncfusion, som er en extension til Visual Studio, der kan bruges til at lave forskellige diagrammer, grafer mm. Med Syncfusion har vi indtil videre formået at danne en graf for hvert produkt, som tilhører et valgt brand (vælges i ComboBox), i samme koordinatsystem.

For at gøre vores koordinatsystem tilpas dynamisk, så brugeren selv kan angive fra hvilket brand de ønsker at se produkter fra, har vi været nødt til at lave en del logik i code-behind-filen. Herved bryder vi helt bevidst vores MVVM's-arkitektur, som i udgangspunktet går ud på, så vidt det er muligt at holde code-behind fri for kode.

Alternativt skulle vi hardcode et maksimalt antal af kurver ind i XAML-filen, hvilket ville være ret forstyrrende for udvikleren, når dette skal vedligeholdes.

Projektstyring

Product backlog

Vi har på baggrund af vores møder med PO, udviklet en liste over ønsker af funktionalitet til systemet. Dette har vi så formuleret til user stories, som vi derefter kan arbejde med. Den vigtigste funktionalitet af vores user stories er derefter blevet prioriteret efter, hvad PO helst ville have med i systemet. Dog er nogle af vores product backlog items blevet sorteret efter, hvad vi mente, der skulle laves, før vi kunne gå i gang med at lave PO's ønsker. Derefter har vi så angivet en størrelse til hver user story angivet med en T-shirt størrelse, som vi har estimeret ud fra den innovative metode - planning-poker, som vi har foretaget i gruppen. Størrelsen angiver, hvor lang tid vi vurderer, at de forskellige implementationer kommer til at tage at udføre. De største user stories (XXL) har vi estimeret til at tage 2 uger, hvor den mindste (S) estimeres til at tage under en arbejdsdag.

Nedenunder ses den Product Backlog, vi er endt med til sidst i projektet. Den er blevet udbygget efter næsten hvert møde med Jacob for at få alle hans ønsker med.

Product backlog items

1. Som bruger vil jeg kunne finde en liste over samtlige produkter af varer på vores lager i det kommende system. **M**
2. Jeg vil som bruger gerne have muligheden for at se, hvornår jeg skal bestille en given vare hjem til lageret. **XL**
3. Jeg vil gerne kunne hente alle vilkårlige salgstal fra csv. filer fra alle foregående år i Powerbanken.dk's levetid. **M**
4. Jeg vil gerne kunne se salgsstatistikker over vilkårlige års salg af varer. **M**
5. Jeg vil ud fra salgsstatistikkerne fra tidligere år, kunne bruge et forecast-system, som skal beregne hvor mange varer, jeg skal bestille hjem på et givent tidspunkt. **XXL**
6. jeg ønsker at når forecast-systemet rammer, en aktiveringsgrænse af produktet nået, skal brugeren have en notifikation af systemet. **S**
7. Jeg ønsker der skal være; en bestillingsdato, en minimumsgrænse for lagerbeholdningen, og en kritisk mængde (når beholdningen går i minus), der indikerer hvornår, jeg skal bestille en given vare hjem for et produkt. **L**
8. Jeg skal kunne bruge systemet til at sammenligne varers indkøbspriser for at finde den samlet billigste indkøbspris.
9. Jeg vil kunne tilføje varer til systemet, når Powerbanken får nye produkter i deres sortiment. **S**
10. Jeg vil som bruger have en valutaomregner, når jeg skal tilføje varer. **M**
11. Jeg vil som bruger gerne kunne aktivere og deaktivere produkterne. **S**
12. Som bruger ønsker jeg en grafisk visualisation af salg på brands. **L**
13. Jeg ønsker som bruger en grafisk visualisation af de enkelte produkters salgstal, så jeg får et overblik. **L**
14. Jeg kunne godt tænke mig at kunne afsende ordrer til leverandørerne direkte fra systemet, så systemet selv kan generere ordrer ud fra de udregnede bestillingsdatoer.

Udvikling af product backlog

Vi valgte at revurdere måden, hvorpå vi havde opstillet vores product backlog items.

Problematikken lå i at vores user stories var for store i forhold til “user-stories” teorien. Vi ville derfor gerne splitte vores user stories op i mindre product backlog items, så vi nemmere kunne vælge en enkelt user story og arbejde ud fra den. Før blev vi f.eks. nødt til at arbejde med oprettelsen af varer i systemet, før vi kunne beregne på, hvornår de ville løbe tør i systemet. (Se product backlog item 1)⁶. Nu da vi har separeret vores user stories, kan vi arbejde mere agilt med et enkelt stykke funktionalitet i systemet.

Framework

Vi har brugt det agile framework; Scrum til at planlægge vores projekt. Vores sprints har hhv. strukket sig over 2 uger med undtagelse af vores pre-sprint og sprint 4, som har haft en varighed på en uge. Vores pre-sprint gik med at have et møde med vores PO, hvor at vi fik en forståelse af virksomheden samt den problemstilling som virksomheden fremlagde, så vi kunne lave de relevante PBI'er. Derudover lagde vi også en overordnet plan for projektet.

Hvert sprint har taget udgangspunkt i de sprint planning meetings, som vi har haft med vores PO. Vi har dog ikke kunne holde et møde med vores PO i to uger grundet, at de har deres højsæson lige nu. Derfor valgte vi at slå sprint planning meeting og sprint review sammen til møderne. Dette gjorde, at vi kunne holde et møde i slutningen af hvert sprint, hvor vi kunne vise, hvad vi havde lavet for derefter at snakke om de udfordringer, vi har haft. Samt at vi kunne planlægge, hvilke PBI'er, vi skulle arbejde med i det efterfølgende sprint, hvis PO havde nogle ændringer i prioriteringen.

Når vi havde fundet de PBI'er, som vi skulle sætte i vores sprint backlog, splittede vi dem først ud i use cases⁷. For at finde de tasks som ville være nødvendige for at opnå vores definition of done, brugte vi innovative metoder⁸ som f.eks. reverse brainstorm. Dette gjorde, at alle i gruppen var inde over processen med at udarbejde tasks. Efter at vi havde fået oprettet de relevante tasks og fundet en definition of done for de enkelte tasks, brugte vi planning poker til at estimere størrelsen samt tiden,

⁶ Se Bilag 5 – Product Backlog Items. Side 55

⁷ Se Bilag 11 – Scrumboard. Side 64

⁸ Se Bilag 6 – Innovative metoder. Side 56

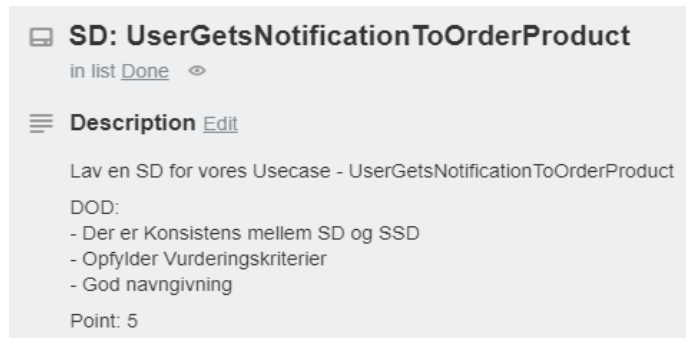
som det ville tage at lave de enkelte tasks. Denne estimering brugte vi også til at vores burndown-chart.

Vores brug af Scrum

Vi startede hver dag med at afholde et daily scrum meeting, hvor vi besluttede, hvilke tasks vi ville arbejde med den givne dag.

Derudover brugte vi også mødet til at snakke, om det arbejde vi havde lavet dagen inden for at sikre os, at de tasks vi havde lavet, opfyldte deres respektive DOD. Til sidst snakkede vi også om der eventuelt ville

være nogen ting, som kunne påvirke arbejdet. Derudover opdaterede vi også burndown chart. Endvidere kan de enkelte tasks, og hvordan vi valgte at uddele dem, samt de beslutninger og overvejelser, som vi har haft under vores daily scrum ses i vores projektlog⁹



Eksempel på Task

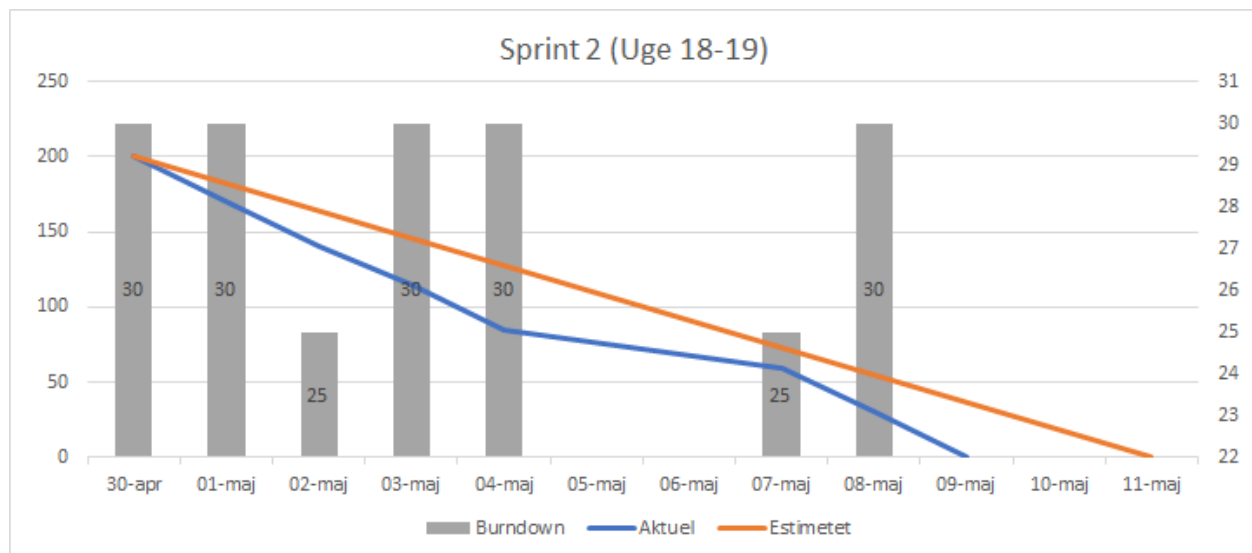
I slutningen af hvert sprint afholdte vi vores samlede sprint review og sprint planning meeting. Der fik vi gennemgået og snakket om arbejdet i det afsluttede sprint, samt det vi ville arbejde med i det efterfølgende sprint. Derefter sluttede vi hvert sprint af med at holde et internt sprint retrospektive for at snakke om, hvordan gruppearbejdet har gået og kunne belyse eventuelle problemer i gruppen så de hurtigere kan løses. Vi lavede også sprint grooming i slutningen af flere sprint efter at havde haft møde med PO, dette gjorde vi for at sikre, at vores PBI stemte overens med, hvad PO ville have.

Burndown-chart

Vi har gjort brug af burndown-chart til at visualisere, hvor langt vi er nået kontra, hvor meget vi mangler at lave, for at blive færdige med vores sprint. Vi har brugt burndown-chart til at kunne beregne arbejds mængden, så vi i starten af hvert sprint kan reflektere over foregående sprints arbejds mængde. Herefter har vi så kunnet lave de nødvendige ændringer som f.eks. at tilføje/fjerne task eller ændre den måde, vi arbejder på, hvis måden, vi arbejder på, ikke er tilfredsstillende.

⁹ Se Bilag 14 – Projektlog. Side

Når vi planlægger vores sprint, og beslutter hvilke tasks vi skal lave for at opnå vores DOD, tildeler vi de forskellige task point, ud fra hvor store de er, og hvor lang tid vi estimerer at de vil tage at



Burndown-chart Sprint 2

lave. Vi har besluttet i gruppen af hvert sprint har 250 point, som vi så deler ud på de forskellige tasks. Dette gjorde vi ved at bruge planning poker. Hver dag, når vi afholder vores daily scrum meeting, opdaterer vi vores burndown-chart og bruger dette som udgangspunkt til at diskutere, hvordan arbejdet skrider frem.

Refleksion over burndown-chart

Brugen af et burndown-chart har hjulpet os med at strukturere vores sprints, da vi ved at give de forskellige tasks en størrelse i form af point har kunne fordele dem ud, så vi som udgangspunkt har lavet lige meget hver dag. Det har også været brugbart da det har givet os et overblik over hvor langt foran eller bagud vi er i vores sprint. Dette har gjort, at vi har kunne lave de nødvendige tiltag tidligere, som f.eks. at ændre arbejdsformen hvis den ikke har fungeret.

Roller

Vi havde i forbindelse med sidste projekt kun en scrum-master i projektet, men læringen og udfordringen ved at være scrum-master, skulle tilfalde os alle i dette projekt. Dette ville gøre os klogere på frameworket og måden at organisere arbejdet på i gruppen. Vi vil bytte scrum-master rollen ud mellem os fra uge til uge, for at se om dette giver et bedre udbytte for arbejdet.

Med henblik på development-teamet inddeler vi os, så alle får lov til at lave forskellige tasks, så det ikke altid er den samme, som skriver kode eller udarbejder artefakter. Dette gør vi med det formål, at læringen udvides og man får en større erfaring med alle aspekter inden for agil udvikling.

Jacob og Christoffer fra powerbanken har ageret PO og har stået for at definere deres problemstilling på en sådan måde, at det har været muligt at lave PBI'er, som de så har prioriteret. Det er også dem, som beslutter, om projektet lever op til deres forventninger, og om der er nogen nye PBI'er, samt om der skal ske en ændring af prioriteringen. Derudover vælger de også, om projektet skal fortsætte, eller om det skal slutte før tid. De har haft til ansvar efter bedste evne at besvare de spørgsmål, som vi i development-teamet har haft.

Når vi har kommunikeret med PO, har det været via E-mail eller ved møder. For hvert møde, vi har afholdt, er det primært scrum-master, der har dialogen med PO, og som sørger for, vi er på samme side, når vi diskuterer problemstillinger og funktionalitet.

Dette gør vi med fokus på dialogen med PO, og hvordan vi formulerer os bedst i forhold til de ikke it-kyndige product owners. Dette giver også det enkelte gruppemedlem lov til at opleve dette.

Risici

Vi er i starten af projektet blevet enige om at lave en gruppekontakt¹⁰ Ved at lave en gruppekontrakt har vi hver især kunne sætte ord på de forventninger, vi har til hinanden og gruppearbejdet, som i fremtiden kan bruges til at løse konflikter. Derudover har vi også udarbejdet en handlingsplan, skulle det uventede skulle ske som f.eks. hvis et medlem bliver syg eller på anden måde ikke kan deltage.

Nedenunder ses en liste af forskellige risici, vi som en gruppe kan komme ud for og hvordan vi eventuelt kan løse dem:

Sygdom i gruppen	Hvis det skulle ske, at en eller flere af gruppens medlemmer skulle blive syge i en længere periode. Ville dette gøre at gruppen ville have en nedsat arbejdskraft, og det ville muligvis have sværere ved at opnå DOD for det pågældende sprint.	Hvis gruppen skulle blive ramt af sygdom, ville de syge gruppemedlemmer være forpligtet til at kommunikere med resten af gruppen og den sygemeldte arbejder hjemmefra i det omfang som er muligt. De fremmødte medlemmer arbejder videre med projektet for at nå i mål. <ul style="list-style-type: none">• Sygdom kan man selvfølge ikke sikre sig imod.
------------------	---	---

¹⁰ Se Bilag 13 – Gruppekontrakt. Side 66

Nedbrud af IT	I tilfælde af at et gruppemedlems computer går i stykker, ville personen muligvis ikke kunne bidrage til arbejdet da personen ikke ville kunne arbejde på kode eller dokumentation.	Da alle i gruppen har en nyere computer, ser vi ikke at dette er sandsynligt. I tilfælde af at dette skulle ske, har alle i gruppen sikret sig at de ville kunne låne eller købe en ny computer så det max ville være en dag de ville være ude af stand til at arbejde på projektet.
Tab af dokumentation	I tilfælde af at noget af den dokumentation som vi laver bliver slettet eller mistes som følge af nedbrud af IT, ville dette kunne sætte gruppen bagud i forhold til vores tidsplan, da det ville være nødvendigt at genskabe det tabte dokumentation.	For at sikre os imod dette, har vi i gruppen aftalt at alt hvad der laves af dokumentation ikke gemmes lokal, men sættes ind i det google-doc vi har så alle har adgang til det og kan se hvilke ændringer der er blevet lavet. Derudover findes der versionsstyring i google-doc, så man kan altid gå tilbage til en ældre version af vores dokumenter. I tilfælde af at det ikke kan sættes ind i Google-doc har vi oprettet en outlook-gruppe hvor at man kan sende dokumenter så alle i gruppen modtager dem.
Tab af Kode	I tilfælde af at noget af den kode vi har skrevet bliver slettet eller ændres i så det ikke længere er brugbart, ville dette kunne sætte gruppen bagud i forhold til vores tidsplan, da det ville være nødvendigt at skrive koden igen.	For at sikre os imod dette bruger vi Github så alle altid har adgang til den seneste kode. Hvis det er nødvendigt, laver vi flere branches så alle kan arbejde på koden uden at det skaber problemer for andre. Dette gør også at hvis man skifter computer pga. Nedbrud af IT hurtigt ville kunne hente alt koden igen.
Bagud i tidsplan	Hvis gruppen af den ene eller den anden grund ikke opnår den DOD som vi har sat os for i et pågældende sprint, ville dette sætte gruppen bagud da det ville være nødvendigt at skyde hele tidsplanen frem, og derfor muligvis ikke ville kunne nå i mål med alle vores PBI	Vi laver som udgangspunkt en arbejdsplan for vores sprint for som skal hjælpe os med at nå i mål med alle vores task, i tilfælde af at vi ikke kan nå det hele, er vi i gruppen indstillet på at arbejde længere hver dag og i værste tilfælde må vi fjerne nogle PBI. da vi laver vores PBI'er efter hvad PO har prioriteret som det vigtigste, ville vi kunne fjerne de mindst vigtige PBI'er.
Mister kontakt til PO	Hvis PO af en eller anden årsag vælger at koble sig fra projektet, eller helt simpelt bare ikke kan være en del af det længere, er det nødvendigt at vi laver en handlingsplan for hvordan vi skal reagere i en sådan situation for at projektet og vores eksamen ikke bare går tabt.	Vi vil selvfølgelig først og fremmest sørge for at tilfredsstille PO så de ikke mister interesse i projektet. Skulle dette alligevel ske, vil vi finde en anden person der kan agere som en ny PO. Enten en ny virksomhed der kan bruge et lignende produkt, eller en vi kender der kan bruges som ny PO.

Dokumentation af projektarbejdet

Begrundelsen for at vi skriver projektlog er, at det skal være en tidsplan, som vi senere vil kunne kigge tilbage på, bl.a. for at se hvilke dage, vi har udført hvilke tasks. Hovedformålet med den er dog, at vi vil skrive vores overvejelser og noter ind hver dag, så vi kan gå tilbage og se, hvorfor vi har taget et bestemt valg. Det skal derfor forstås som noter til os selv, da det er umuligt at huske, hvad der er blevet lavet hver dag ellers.

Derudover bruger vi det også til vores daglige scrum-meeting, både til noter, men også til de forskellige tasks til hver enkelt dag, for at vi som sagt kan kigge tilbage og se hvilke tasks der er blevet udført på hvilke dage. De enkelte tasks er naturligvis taget ud fra vores scrumboard, som vi har brugt flittigt gennem udviklingsprocessen.

Eksperimenter

Vi har i løbet af vores projekt besluttet at vi vil benytte os af eksperimenter. Vi har valgt at foretage forskellige eksperimenter, dels fordi vi har overskud til det og dels fordi det kan give konstruktiv feedback til vores projekt og gruppearbejde. Især under eksperimentet: Brugervenlighedstest fik vi også et godt indblik i, hvordan vores brugere, ville have opbygget systemet, og det hjalp os meget med opbygningen af designet. Det er dybere udforsket i nedenstående afsnit.

Kommunikation i gruppen

Vores hypotese til eksperimentet lød på:

Fungerer kommunikationen og arbejdsmængden bedre når vi alle sidder og arbejder sammen i samme lokale?

Resultat af eksperiment:

Udførelsen af eksperimentet har haft en meget negativ effekt på både arbejdsmængden, men også først og fremmest kvaliteten af vores tasks. Vi kunne ud fra en re-evaluering af de tasks, vi havde fået lavet, konkludere at de fleste skulle refaktoreres og/eller omskrives. Et eksempel på en task var, at et gruppemedlem skulle udarbejde en række vurderingskriterier for vurderingen af rapporten, men arbejdet blev ikke fyldestgørende nok.

Dette blev også tydeligt udforsket, da vi dagen efter snakkede om dem, og der kom mange flere kriterier på. Dette er også den primære konklusion på eksperimentet; Vi får lavet et stykke arbejde,

men det bliver ikke særligt fyldestgørende, da kommunikationen er limiteret. Arbejdsmængden er ligeledes heller ikke den samme som, når vi sidder samlet og kan holde hinanden op på, at vi skal opnå det bedst mulige resultat hele tiden.

Vi havde som udgangspunkt prøvet at måle tiden på hvor meget arbejde, vi fik lavet derhjemme på den normale arbejdstid, kontra hvor meget vi får lavet i gruppen, når vi sidder samlet.

Resultatet blev sådan cirka at: Tiden for fokuseret arbejde blev beskåret med $\frac{2}{3}$ i forhold til samarbejde i gruppen.

Brugervenlighedstest

Vores hypotese til brugervenlighedstest lød på:

Hvor godt kan PO bruge systemet (samt hvor hurtigt), lever systemet op til PO's ønsker og gavner systemet i forhold til før?

Resultat af brugervenlighedstest:

For at sikre at vores system lever op til de kriterier, som vores PO har, og at designet er brugervenligt, har vi haft udført en brugervenlighedstest. Vi udførte testen i uge 20, da vi på daværende tidspunkt havde fået lavet de primære funktioner PO ønskede på en sådan måde, at det nu var muligt at teste på både navigation og brugervenlighed.

Det, vi testede for, var, om vores design var logisk, så PO selv ville kunne finde rundt i systemet. Dette testede vi ved at give vores PO nogle opgaver, han skulle løse i systemet, og han skulle derefter besvare nogle spørgsmål omkring, hvilke tanker han havde vedrørende systemet.

Det var meningen at begge vores PO's skulle prøve en gennemgang af systemet, men da den ene ikke var tilgængelig, da testen skulle udføres, var dette ikke muligt.

Det vi fik ud af brugervenlighedstesten var, at der var nogle problemer med vores design, som blev belyst, såsom der skulle være flere attributter i vores datagrid og ligeledes i vinduet, hvor man tilføjer produkter. Derudover var der også nogle ændringer vedrørende funktionaliteten, som vores PO ønskede, som vi så yderligere kunne arbejde på.

(Se bilag for den fulde test plan og resultater.)¹¹

¹¹ Se Bilag 8 – Brugervenlighedstest. Side 58

Fremtid efter overdragelsen af projektet

Når vi i vores sidste sprint i uge 22 har afleveret vores projekt, har vi i samarbejde med PO, allerede snakket om fremtidig funktionalitet i systemet.

Tidligt i projektets start, så de gerne at systemet skulle kunne implementeres på Mac og lave en GTK eller en web-applikation til dem. Dette ville vi gerne bruge tid på efter eksamensforløbet er slut, hvis de er tilfredse med systemets funktionalitet i WPF. Efter vi er færdige med eksamensforløbet, vil vi også have friheden til at fokusere mere på det grafiske design og visning af grafer, som PO kraftigt har ønsket. Derfor ville vi gerne kunne give dem muligheden for at, filtrere bedre i produkterne, end de kan nu.

Noget andet, som man kunne arbejde videre på, er at få vores løsning til at kunne bruge Magentos API, så salgstal kan hentes automatisk fra PowerBanken.dk's webshop-løsning. Dette vil gøre det meget nemmere, da man ikke manuelt vil skulle eksportere månedlige salgstal fra Magento for derefter at importere dem i vores system via csv.-filer. Denne operation vil i stedet kunne foregå helt automatisk. En anden fordel vil være, at systemets beregninger af ordredatoer sandsynligvis bliver mere nøjagtige, eftersom den data, der vil blive brugt til beregningerne, samt lagerbeholdningen er helt korrekt.

Handlingsplan for videreudvikling

Da vi ikke regner med at Powerbanken.dk kommer til at gøre brug af vores system, har vi dog stadig taget os de overvejelser, hvad der skulle ske, hvis de gjorde det.

Vi ville give dem nummeret til en kontaktperson i gruppen, som de ville kunne ringe til, hvis systemet ikke skulle fungere, som det plejer. Derudover ville vi også regelmæssigt have et møde med PO hvert halve år, for at lave back-up af deres data. Dette interval er sat efter, at Powerbanken.dk ikke så ofte opretter nye varer i deres system. Derfor ville tabet af data ikke være så stort, hvis vi allerede fra starten laver en back-up.

Et stort problemområde for dette system er, at databasen lige nu er EAL's database, og de ikke ville kunne bruge den i længden. Dog ville vi gerne have dem til at skifte over til en lokal database, såsom SQLite. Dette ville også gøre at, hvis forbindelsen til netværket ikke eksisterede, så ville programmet stadig kunne fungere.

Vi har også diskuteret rettighederne vedrørende kildekoden, og om man senere hen skulle forhandle om en eventuel viderebygning på systemet. Dette kunne f.eks. være Mac-konverteringen der kunne

blive konkret her. Ellers kunne der også være tale om eventuelle opdateringer, hvis f.eks. Børsens hjemmeside blev opdateret, så ville vores valutaomregner ikke fungere optimalt længere. Dette skal vi skal dog i dialog med Powerbanken.dk omkring, hvis de ønsker en videreudvikling på funktionalitet i systemet.

Konklusion

Som vi når slutningen af vores eksamensprojekt, falder de sidste brikker på plads og vi afrunder så småt de sidste tasks. Ved slutningen af vores projekt, er der forskellige ting, der er gået bedre end andre, heriblandt implementationen af SyncFusion og den begrænsede kontakt med PO, bare for at nævne et par stykker. Det har dog ikke kun gået skidt igennem hele projektet, og der har været en masse personlig udvikling under hele projektet. Blandt andet er vi alle blevet meget bedre til koordination af arbejde, kommunikation internt og eksternt, og til programmering, da vi igennem hele projektet for det meste har arbejdet på at udfordre os selv og lære.

Selve programmet er næsten blevet som PO ønskede, bortset fra de åbenlyse mangler, såsom at de ønskede en Mac-løsning og at vi ikke rent visuelt har opfyldt alle deres ønsker vedrørende graferne. Vi mener dog, rent funktionsmæssigt, at vi har fået løst vores PO's problemstilling og lavet et program de i teorien kunne tage i brug i morgen. Med henblik på rapporten har vi fulgt EAL's datamatiker formalia, kvalitetskriterier og de generelle rapportskrivningsretningslinjer - som vi mener har været med til, at denne rapport lever op til de krav, som er stillet af vores undervisere og selvfølgelig PO.

Litteraturliste

- “Applying UML and Patterns”, 3rd ed., Craig Larman, Prentice Hall, 2004
- “Professional Test-Driven Development with C#”, J.Bender & J.McWherter, Wrox, 2011
- “Database Concepts”, 7th ed., David M. Kroenke, David J. Auer, Pearson, 2015
- På Sporet af Det Gode System
- [https://fronter.com/eal/links/files.phtml/1766985795\\$277013695\\$/1.+Semester/Systemudvikling/Diverse+litteratur/27creativityinnovationtools.pdf](https://fronter.com/eal/links/files.phtml/1766985795$277013695$/1.+Semester/Systemudvikling/Diverse+litteratur/27creativityinnovationtools.pdf)
- <https://www.usability.gov/how-to-and-tools/methods/planning-usability-testing.html>

Bilag

Bilag 1 – Usecases

Use case - CalculateOrdersBasedOnForecast

Jacob vil gerne have at vide, hvor mange produkter af en Anker-powerbank han skal bestille hjem i juni måned. Han vælger derfor en given måned og en procentsats i systemets forecast-funktionalitet. Herefter får han en samlet liste over varer med en beregning over hvor mange varer, de regner med at sælge.

Hoved succes scenarie : Jacob modtager en liste fra systemet med samtlige varer fra en given måned og deres forventede vækst for næste års måned.

Alternative scenarier :

- Jacob vælger ikke en måned og modtager en fejl, da systemet kræver en valgt måned.
- Jacob indtaster andre karakterer end tal ind i procent-afvigelsen, og kan ikke få lov at beregne den forventede vækst.
- Systemet har ikke den valgte csv. Fil importeret, og får derfor ikke noget output fra den valgte måned.

Use case - ImportProductSalesFromCSV

Mainscenarie

Jacob vil gerne importere månedlige salgstal for samtlige af forretningens produkter, som ligger i en CSV-fil. Han åbner systemet og vælger den fil, han ønsker at importere. Efter han har klikket importer, viser systemet en liste med de importerede data. Nu kan Jacob vælge at gemme disse data i systemet og tilrette den solgte mængde, hvis der er fejl, før han gemmer.

Alternative scenarier

- Den valgte fil indeholder data i forkert rækkefølge eller forkert format, så listen med importeret data vil være tom, efter han har valgt importer.
- Han modtager fejl, hvis han vælger at gemme salgsdata for et produkt for en måned, hvis der allerede eksisterer salgsdata i databasen for samme produkt og periode.

Use case - OrderProductBasedOnGrowthAndDateOfTheYear:

· Jacob vil gerne vide, hvornår han næste gang skal bestille en aktiv vare hjem på baggrund af en vækstperiode, et år, som han indberetter vækst for. Jacob angiver en vækst for det kommende år og herefter beregner systemet en bestillingsdato for alle produkter. Hvis en af bestillingsdatoerne ligger inden for nærmeste fremtid, modtager han en notifikation.

Hoved succes scenarie

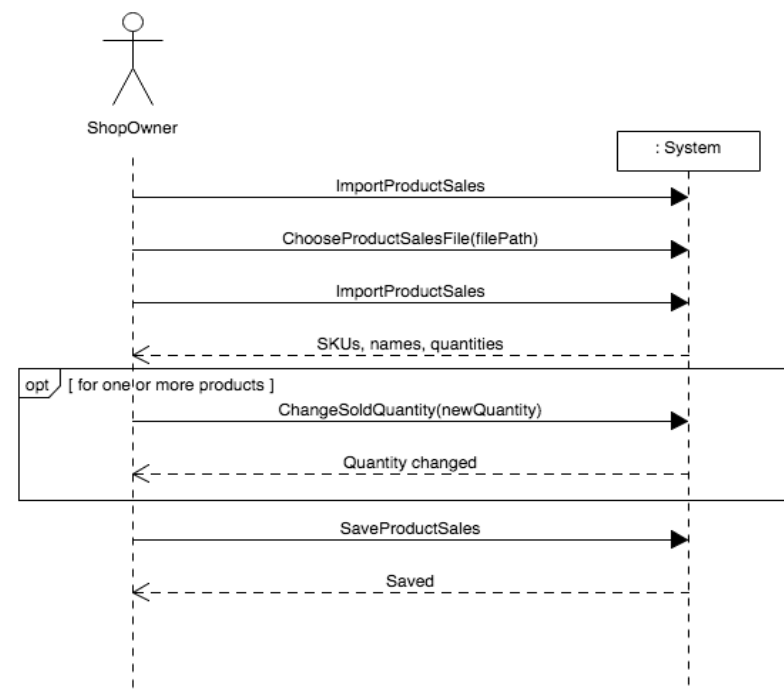
- Jacob får en oversigt over det daglige salgstal og en foreslået ordredato til hvornår, han skal bestille en vare hjem til lageret.

Alternative scenarier

- Hvis Jacob har valgt en dag på året, som allerede er gået. Dette vil skabe en undtagelse i systemet, og informere brugeren om, datoen ikke kan vælges.
- Hvis Jacob vælger en inaktiv vare i systemet, og vil ikke modtage noget information fra den valgte vare.
- Hvis Jacob vælger en vækstperiode for et år, som ikke har noget tidligere data, vil han modtage en fejl.
- Hvis systemet opdager en fejl i den eksterne database

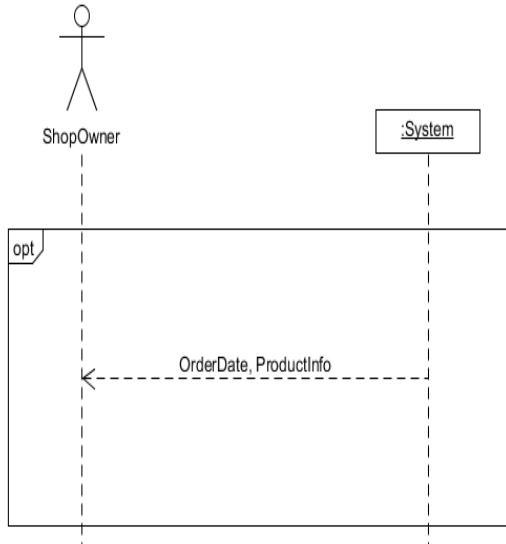
Bilag 2 – SSD'er

Import salg – SSD



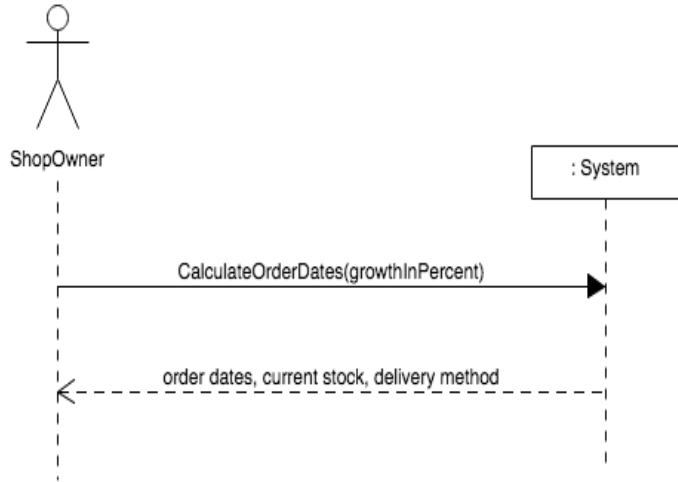
Figur 1

GetNotification – SSD



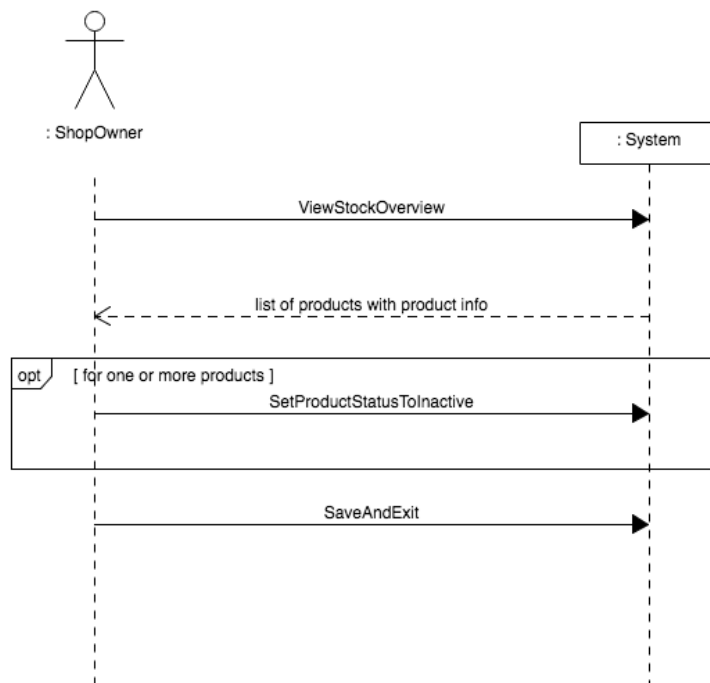
Figur 3

CalculateOrder - SSD



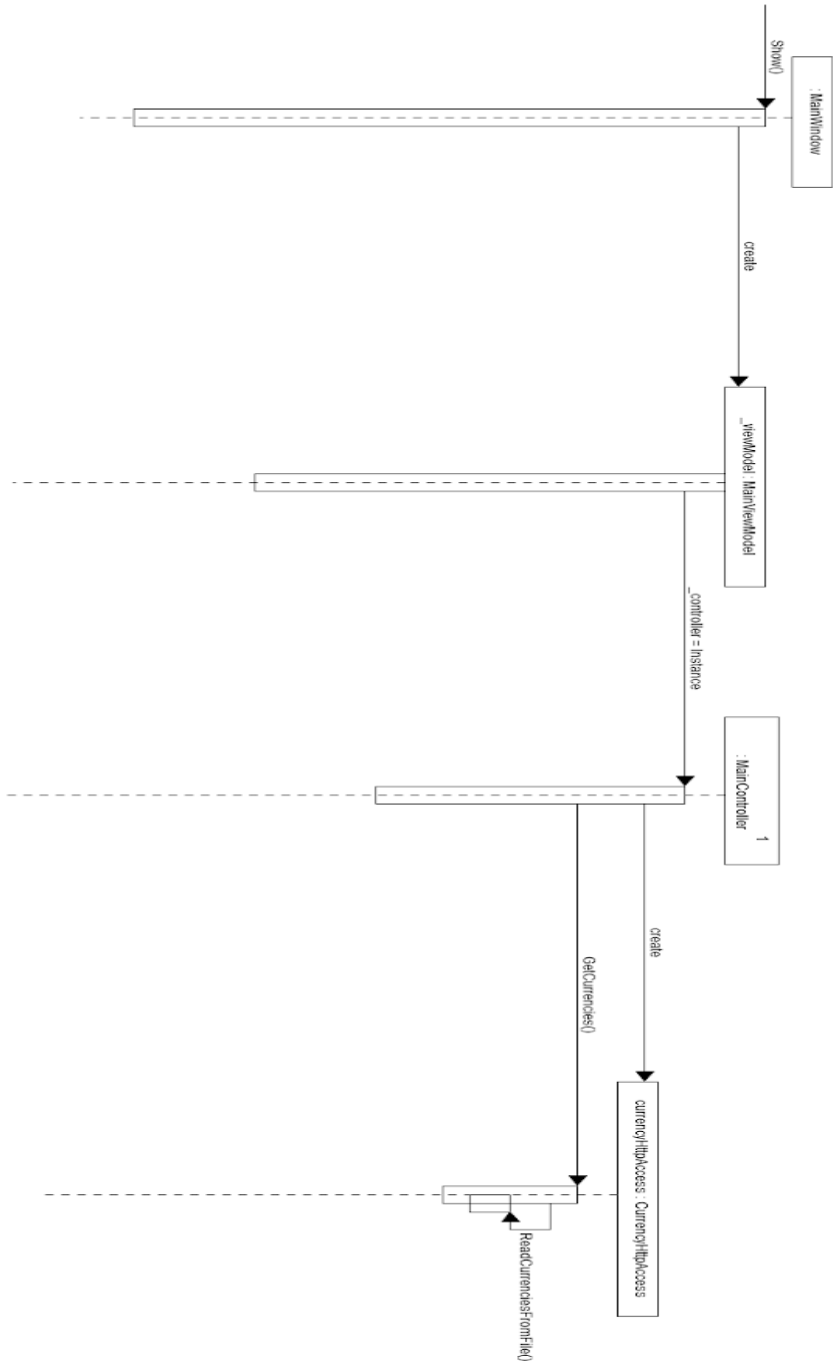
Figur 2

SSD - ViewStock

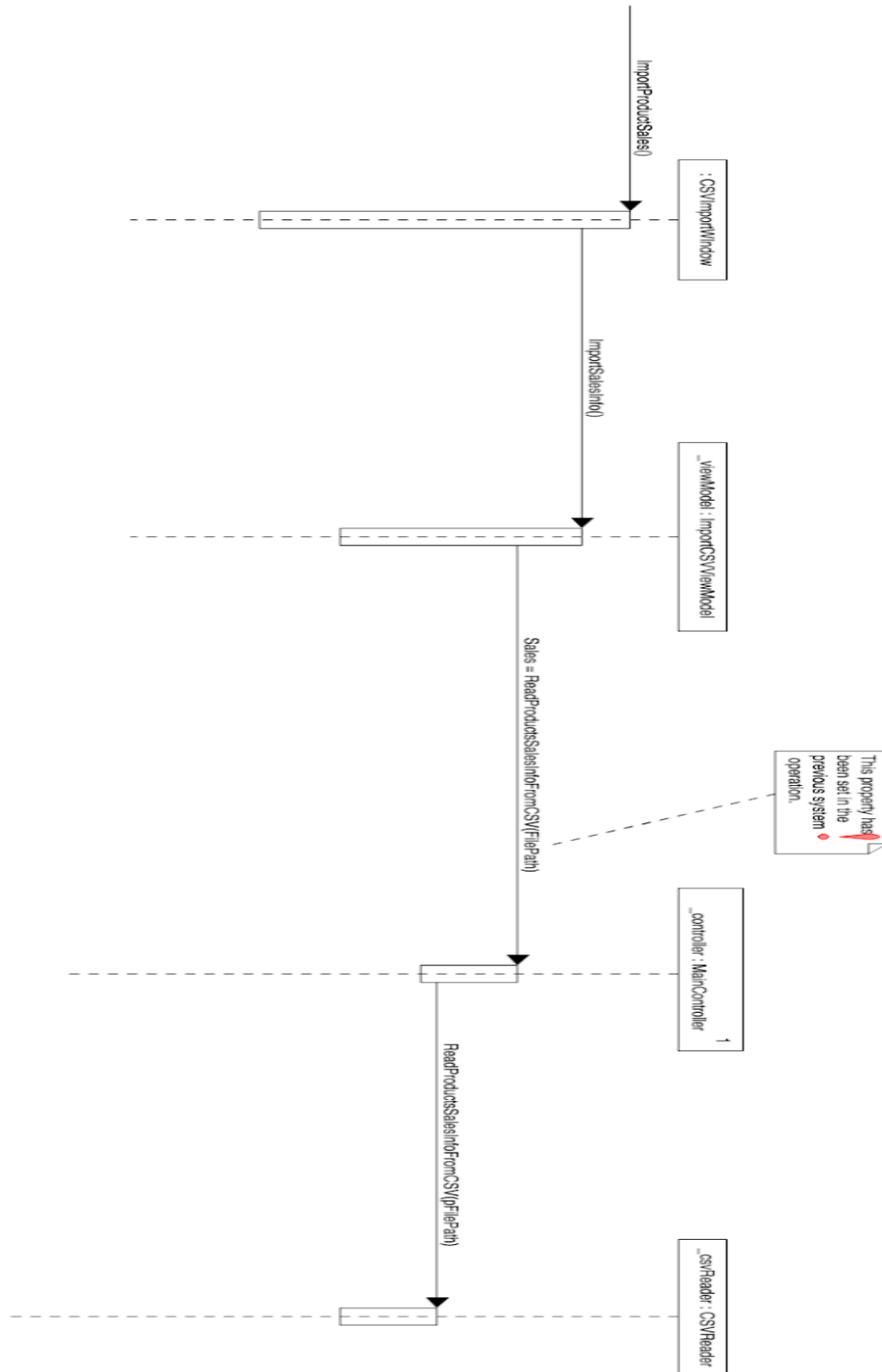


Figur 4

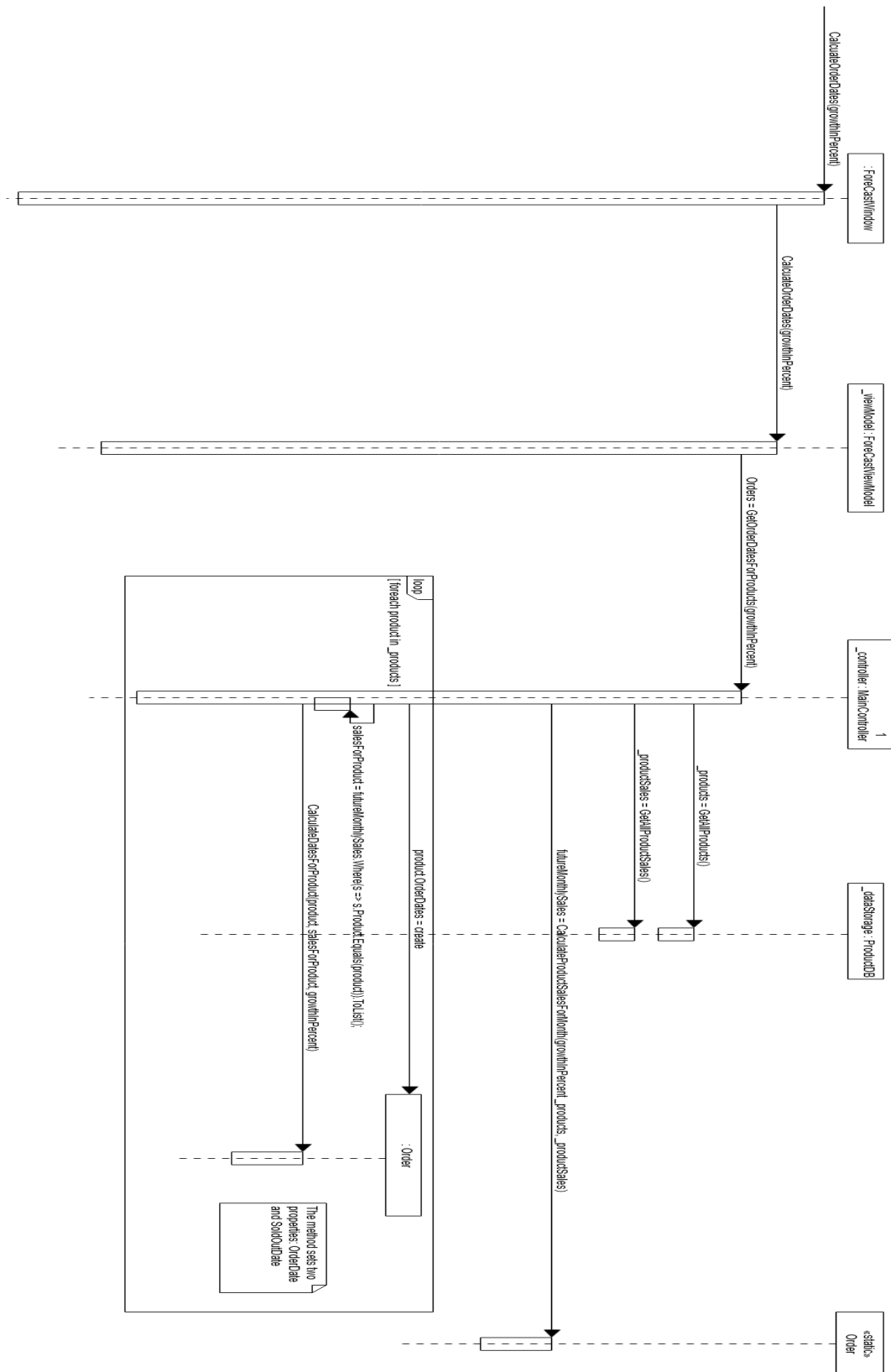
Bilag 3 SD
SD - GetCurrencies



SD - ReadProductSalesFromCSV



SD – CalculateOrders



Bilag 4 - SOC

Operation: CalculateOrderDate(startDate, endDate, expGrowth)

Cross reference: *OrderProductBasedOnGrowthAndDateOfTheYear*

Pre-conditions:

- System has info about all previous years sale
- Information about the sale period and the growth is entered by the user (startDate, endDate, expGrowth).

Post-Conditions:

- A suggested order has been created in the system and shown to the user

Bilag 5 - Product backlog item)

Størrelse: (xxl, xl, l, m, s,)

Prioriteret rækkefølge

1. PO vil kunne bruge systemet til at se hvilke varer, de vil løbe tør for, hvis de ikke køber nye varer ind snart. **XXL**
2. PO vil kunne se hvor mange af hvilke varer de har på lager på nuværende tidspunkt **S**
3. PO vil se et "forecast" over hvor mange varer de forventer at sælge fremtidigt, baseret på forventet vækst, som de selv indberetter i procent, samt forrige salg (årligt?) **XL**
4. PO vil kunne se nogle statistikker over forrige salg, med diagram over salg fra hver uge. **M**
5. PO vil kunne tilføje produkter til kartoteket, og vil samtidigt også kunne slette udgåede varer (Eller sætte dem i en anden kategori, der deaktiverer et produkt, hvis det skulle bestilles på et andet tidspunkt) **M**
6. Vil kunne se hvad der giver mest mening at købe? (Nogle varer kan kun købes en masse, bestilling = tid/antal) **L**

Bilag 6 - Innovations metoder

Da vi i starten af projektet skulle udarbejde mulige løsningsforslag, gjorde vi bruge af nogle af de innovative metoder som vi har haft lært i løbet af vores 1. År. For at på bedst mulig måde kunne komme flere forskellige. Vi brugte klassisk brainstorm såvel som reverse brainstorm. Da vi på den måde kunne se projektet fra forskellige vinkler og skabe ideer til løsninger.

Vi brugte derudover også 100 euro test, som vi ændrede til at beløbet var 250 point. Dette besluttede vi i gruppen da vi i starten af hvert af vores 4 sprint brugte 250 point til at oprette og fordele point til de tasks som vi skulle lave i de forskellige sprints. Og vi ændrede det, da det for os passede bedre fordi det så ville svare til at vi igennem hele projektet har været igennem 1000 point.

Bilag 7 – KPI'er

KPI – Varebestilling

KPI	Vi vil måle hvor præcis vores beregninger omkring dato for varebestilling er.
Hvorfor måles?	Vi måler for at finde ud af hvor meget vores beregninger afvigere fra den præcise dato for bestilling.
Hvordan måles	Vi måler ved at se på hvilken dato PO bestiller vare hjem, kontra hvilken dato vores system har beregnet.
Hvem er ansvarlig for måling?	Målingen vil blive foretaget af hele gruppen.
Forventet målingsdato	Vi forventet at målingen skal laves efter at de har taget systemet i brug.
Forventet værdiinterval for måling	forventet værdiinterval er en afvigelse på +- 5 dage fra den aktuelle dato PO bestiller på
Måling	Ikke foretaget endnu
Handlingsplan i fald målingen ligger udenfor forventet interval	Hvis målingen ligger uden for intervallet, er vores handlingsplan at refakturere den algoritme programmet bruger for på den måde at opnå vores ønskede interval
Ansvarlig for handlingsplan	Handlingsplanen vil blive lavet af hele gruppen.

KPI – Estimeret salg

KPI	Vi vil måle hvor præcis vores estimeret salg er kontra det konkrete salg for en udvalgt måned
Hvorfor måles?	Vi måler for at finde ud af hvor stor en %-vis afvigelse der i forhold til hvor meget systemet estimere vil blive solgt i en udvalgt måned i forhold til hvad der faktisk bliver solgt
Hvordan måles	Vi måler ved at se hvad det aktuelle salg i en måned er og beregner hvor stor en %-vis afvigelser der er i forhold til systemers estimat
Hvem er ansvarlig for måling?	Målingen vil blive foretaget af hele gruppen.
Forventet målingsdato	Vi forventet at målingen skal laves efter at de har taget systemet i brug.
Forventet værdiinterval for måling	forventet værdiinterval er en afvigelse 10-15%
Måling	Ikke foretaget endnu
Handlingsplan i fald målingen ligger udenfor forventet interval	Hvis målingen ligger uden for intervallet, er vores handlingsplan at refakturere den algoritme programmet bruger for på den måde at opnå vores ønskede interval
Ansvarlig for handlingsplan	Handlingsplanen vil blive lavet af hele gruppen.

Bilag 8 – Brugervenlighedstest

Testplan for brugervenlighedstest

Scope: Det, som vi har tænkt os at teste, er alle de funktionaliteter, som vores system har i starten af sprint 3. Vi vil teste dem både som individuelle funktioner samt teste hele systemet som helhed.

Purpose: formålet med testen er at få en forståelse for, om det design, vi har lavet til systemet, er lavet på en sådan måde, så vores PO vil kunne manøvrere i systemet på egen hånd og vil kunne bruge de forskellige funktionaliteter. Samt om designet er lavet på en måde, som PO ser som logisk.

Schedule & Location: Testen vil foregå ude hos powerbanken og vil blive afholdt d.16 maj 2018, hvor at vi også holder sprint demo.

Sessions: Testen vil bestå af 2 sessioner af 15-20 min. Herefter vil der være en pause på 2-3 minutter, hvor vi genstarter programmet, og hører testpersonens respons på systemet. Da vi har 2 testpersoner, vil de hver gennemgå en session.

Equipment: Vi skal bruge en computer med Visual studio med systemets solution-fil, som testen skal foregå på. Den skal også have adgang til internettet, så den kan bruge VPN. Dette er nødvendigt, da vi skal være på EAL's netværk for at kunne bruge databasen. Vi kan bruge Powerbankens internet, men hvis dette skulle vise sig ikke at være muligt, kan vi lave et hotspot via mobil.

Participants: Testen skal udføres af 2 af vores PO's, da det er dem, som kommer til at bruge systemet.

Scenarios: hver session kommer til at bestå af følgende.

- Opret vare
- Søg efter vare/deaktiver vare
- Ændre indstillinger
- Import CSV-fil
- Kør beregning
- Vis grafer over salg.

Successful Task Completion:

1. **Opret vare** - testen er færdig, når de har oprettet en vare.
2. **Søg efter vare/deaktiver vare** - testen er færdig, når de får søgt den vare frem, som vi har bedt dem om, og de har deaktiveret den.
3. **Ændre indstillinger** - testen er færdig, når de får ændret indstillingerne for hvor mange dage frem, de ønsker at få underretninger om bestillingsdatoer.
4. **Import CSV-fil** - Testen er færdig, når de har importeret CSV-filen, og den bliver vist.

5. **Kør beregning** - Testen er færdig, når de har kørt beregningen og får vist det ønskede resultat.
6. **Se statistik over 3 varer** - Testen er færdig, når de har fundet 3 varer på checklisten, og får det viste resultat.

Det data, som vi vil måle, er:

- Hvor lang tid tager det dem at komme igennem de enkelte test.
- Hvor mange gange klikker de forkert.
- Om systemet crasher, og hvis det gør, så hvornår og hvor mange gange dette sker
- Om de forstår, hvad ændringerne i **[Indstillinger]** gør
- Hvor mange gange de bliver nødt til at spørge om hjælp.

Likes, Dislikes and Recommendations

Efter hver test vil vi spørge dem om:

- Hvad, de synes, er godt
- Hvad, de synes, er dårligt
- måden, vi har designet det på, er tilfredsstillende, om de eventuelt har nogle forslag til forbedringer/ændringer.

Efter at de er færdige, vil vi spørge dem om:

- Deres mening om designet som helhed.
- Den måde, vi har designet det på, er tilfredsstillende, og om de eventuel har nogle forslag til forbedringer/ændringer.

Roller:

De roller som vi har til testen er:

- En opstiller test miljøet. - Jakob
- En noterer test-data - Daniel
- En hjælper testpersonerne hvis det er nødvendigt. - Jacob
- En stiller spørgsmål efter hver test. – Simon

Brugervenlighedstest – Resultater og iagttagelser

Tid og iagttagelser:

- **Opret vare**
 - tid: 30 sek.
 - Fandt hurtigt frem til menuitems i toppen og fandt det rigtige punkt, "Filer".
- **Søg efter vare/deaktiver vare**
 - tid: 30 sek.
 - øjeblikkelig respons i resultatfeltet ved tastetryk var en god ting
- **Ændre indstillinger**
 - tid: 10 sek
 - Krævede naturligvis en forklaring, men testpersonen var positiv over for disse funktioner
- **Import CSV-fil**
 - tid: et par minutter
 - Gik hurtigt og nemt – intet at bemærke.
- **Kør beregning**
 - tid: et par minutter
 - Vi var nødt til at forklare hensigten med funktionen, men ellers gik det hurtigt og nemt.
- **Vis grafer over salg.**
 - tid: et par minutter
 - Krævede også lidt forklaring, eftersom vi ikke havde noget tekst, som beskrev om det var tidligere eller fremtidige salg.

Spørgsmål:

- **Hvad er godt?**
 - Tilføj produkt: data, som skal indtastes stemmer fint overens med de informationer, de har om varerne.
 - Søgning: fin oversigt over varer
 - Importer: er gået fint, og man har et godt overblik.
 - Bestillingsdatoer: fint
- **Hvad er dårligt eller trænger til forbedringer?**
 - Tilføj produkt: Valuta skal fremgå af tilføjelsesvinduet. Alternativt skal man kunne vælge, hvilken valuta det indtastede tal skal være i. Leadtime skal også have en enhed (timer, dage, år).
 - Søg: Knap til deaktiverede og aktiverede varer skal skifte tekst ved klik. Søgefeltet mangler en label. Man skal gerne kunne søge på type og brand
 - Bestillingsdatoer: Datoer for hvornår man når nul, forventet bestillingsdato, kritisk lager ved overskridelse af bestillingstidspunkt.
 - Hjælp-knappen er inaktiv
 - En varer indeholder både en salgspris og en indkøbspris
 - Grafer: man skal gerne kunne sammenligne to perioder og angive et tidsinterval, som man vil se en kurve for.
- Man skal kunne oprette salgsordrer/bestillinger ud fra listen af produkter, som de snart løber tør for.

Bilag 9 - Handlingsplan

Siden vores møde med PO i sprint 1, har vi ikke haft kontakt. Vi har prøvet at kontakte dem via den mail vi blev udleveret. Efter næsten 3 uger, har vi ikke haft nogen kontakt, og derfor ikke haft mulighed for at lave nye møder, hvor vi aftaler og viser hvad vi har lavet. Det har haft de konsekvenser, at det har været svært at lave en sprint planning fase, hvor vi bliver enige om hvad vi skal gøre i dette sprint, baseret på PO's ønsker.

Vi vil selvfølgelig fortsætte med at sende dem emails, og skulle problemet være ved uden nogen kontakt via mail, må vi henvende os personligt, da vi ikke har fået udleveret noget telefonnummer. Fra PO's synspunkt, er vi forståelige over at det kan være svært at overskue at bruge tid og kræfter på at hjælpe os, med et projekt de højst sandsynligt aldrig vil tage i brug, da de stadig ønsker en Mac løsning. Ud fra vores sidste møde, kan det også have virket afkræftende at vores GUI har været manglende, da det indirekte har været et ønske fra PO, at systemet har flere visuelle ligheder med programmer fra Mac. PO har dog ikke sagt at GUI skulle være af et hvis grafisk niveau, da vi gjorde dem klar over at GUI var noget af det sidste vi ville arbejde med.

Vores sidste møde med PO, var dog en relativ stor succes, da vi fik besvaret en masse spørgsmål. Det var dog ikke helt uden problemer og det var lidt åbenlyst at de selv havde svært med at huske hvad de ønskede og hvorfor, hvilket kunne resultere i nogle mindre uenigheder blandt PO selv (Jacob og Christoffer), da de begge har forskellige ideer til hvordan systemet skal se ud og fungere.

Jacob; som er vores primære PO og firmaets "marketing guru", forstår at vores projekt er pt. Kun et eksamensprojekt, og det derfor mindre vigtigt for ham at bruge tid på specielle funktioner, som implementering af grafer osv. Generelt ønsker Jacob bare at vi går med hvad vi selv føler for, dog holder os inden for de generelle ideer han har præsenteret.

Christoffer; som altid holder møderne sammen med Jacob er firmaets "Business to business/B2B aktør", og han har derfor en stor indsigt i handel og kontakt mellem firmaerne, så rent naturligt ved han derfor også hvad han vil have. Hvor Jacob siger det kunne være rart med diverse funktioner, kan Christoffer godt finde på at specificere og indblande hans egne ideer til systemet, samt hvordan hvilke tal der skal bruges i graferne, hvilke informationer der skal være tilgængelige osv., hvortil

Jacob bare nikker “ja”. Christoffer fungere derfor som en mere pålidelig PO, da han automatisk tager ansvar over hvad der skal laves og forventer kun et virkende resultat. Desværre er det ikke ham vi kontakter per. Mail.

Hvis vi ikke kan få fat på PO, må vi acceptere at projektet nu ligger i vores egne hænder og vi må derfor selv agere som PO for vores eget projekt. Det eneste vi eventuelt kan gøre, for at simulere manglende PO er at tage mere fat i vores undervisere og vores projekt styringsgruppe.

Konsekvenserne bliver derfor, skulle PO vælge aldrig at svare, et projekt uden en rigtig styre hånd, hvor de retninger vi tager projektet bliver vores helt egne, som på visse måder begrænser projektets scope og plan, da vi selv kan sige “det der kan vi finde ud af” og modsat.

16-05-2018

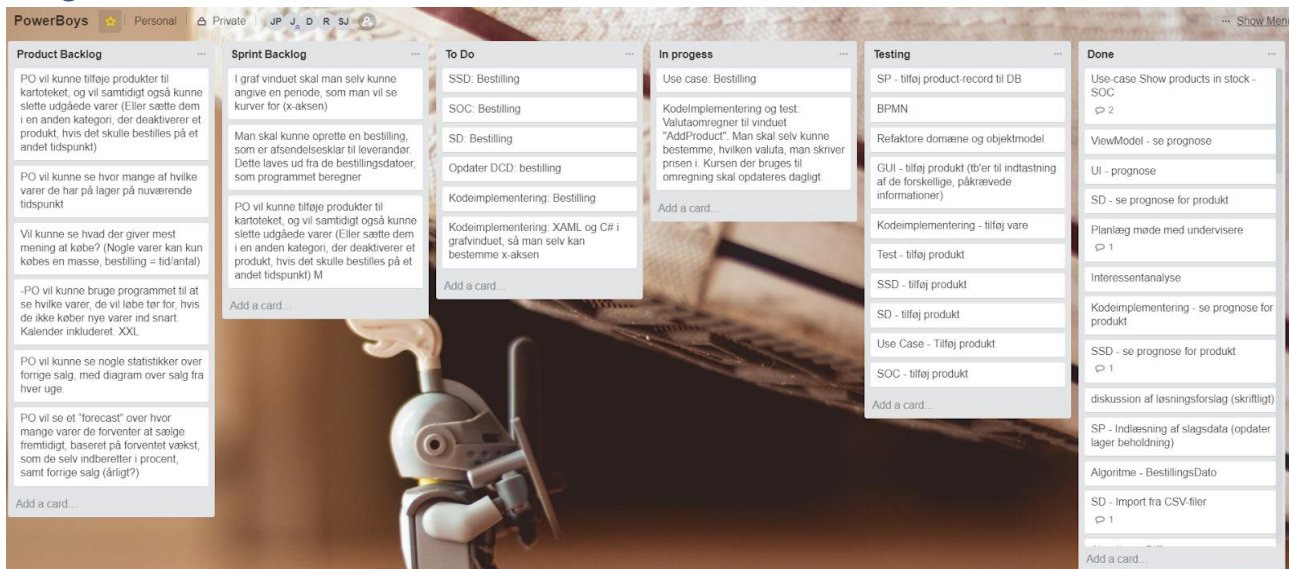
PO har igen vendt tilbage, og vi har besluttet at vi vil forbedre vores GUI for at gøre systemet lidt mere præsentabel. Til mødet foretog vi en brugervenlighedstest, med udmærket resultater. Til testen testede vi blandt andet hvor godt Jacob kunne manøvrere rundt i systemet, slette og tilføje vare, samt at udforske nogle af de muligheder vi har lavet, såsom grapher, notifikationer ved bestillingsdato m.m.

Bilag 10 – kommunikationsplan

HVEM - hvilken interessant er målgruppe for kommunikati on?	HVAD - Hvad er budskabet ?	HVORDAN - Hvilke kommunikations midler anvendes?	HVILKEN (effekt) - hvad skal opnås, at interessenten gør som følge af kommunikati on?	HVORNÅR - Specificer ift. Projektets milepæle og tidsplan	ANSVAR - hvem har ansvar for, at det sker?
Product Owners	Product owners skal opdateres omkring udviklingsprocessen. Deres løbende feedback og accept er nødvendig for at vores arbejde kan fortsætte. De er med andre ord vores vigtigste interessant.	Fysiske møder, Skype og E-mail	I sidste ende er det PO, som giver os arbejdsopgaver. Derfor skal vi gerne have noget input, som vi skal bruge til senere sprints. Omvendt set er formålet også, at de får en idé om, hvordan udviklingen skrider frem.	Der vil være løbende kontakt via mail og skype efter behov, måske 2-3 gange i ugen. Møderne vil foregå planlagt én gang pr. sprint.	Det har vi som udviklergruppe. Derfor er det vigtigt, at vi får aftalt møderne rettidigt, så vores kommunikationsplan ikke går i vasken.
Projektledere	De skal hjælpe med at motivere os og holde os op på vores plan.	Foregår via møder eller via e-mail.	Som udgangspunkt hjælper det gruppen med deres eksamen.	Som minimum 1 gang hver anden uge. Det ville måske være relevant at holde lidt flere møder senere i projektet.	Primært projektlederne selv.

Projektledelse (lære)	Skal hjælpe os med at holde os indenfor vores “rammer”, samt have et mere “professionelt” synspunkt på vores arbejde, og rådgivning.	Møder, aftalt på forhånd.	At vi ikke slår for stort brød op, og holder os inden for pensum, samt eventuelt nødvendig rådgivning.	Fast ugentlig vejledningsmøde, (30 min.) dog mulighed for vejledning efter behov.	Det har vi som projekt gruppe selv ansvar for.
-----------------------	--	---------------------------	--	---	--

Bilag 11 - Scrumboard



<https://trello.com/b/kiHZD4sz/powerboys>

Vi har igen taget brug af Trello, da vi ikke har endnu, har sat os ind i Githubs Scrumboard funktion.

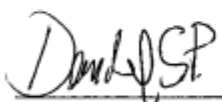
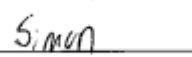
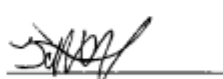


Bilag 12 - Notifikationsvindue
- brugervejledning



2 nye varer skal bestilles inden for de næste 7 dage - se vinduet med bestillingsdatoer.

OK

Bilag 13 – Gruppe Kontrakt

Gruppe Kontrakt	Dato: 09/04/2018	
Denne gruppekontrakt er gældende for følgende gruppemedlemmer.		
<ul style="list-style-type: none">• Jakob Vingaard Petersen• Simon Egebjerg Jensen• Daniel Stuhr Petersen• Rasmus Bak Petersen• Jacob Lau Petersen		
§1 - alle involverede parter skal udvise engagement på alle tidspunkter		
§2 - hvis en eller flere parter ikke formår at møde op på en given dag, bliver der fra gruppen påkrævet sanktioner.		
§2a - hvis en part ikke er i stand til at møde op eller på anden måde ikke deltage i projektet, har denne part pligt til at formidle dette til de resterende parter		
§2aa - Hvis fraværende part ikke giver besked til gruppen, bliver der fra gruppen påkrævet sanktioner.		
§2b - Alle beslutninger bliver truffet af de fremmødte parter.		
§3 - Det forventes, at hver part har forberedt sig tilstrækkeligt til den givne dag.		
§4 - Hvis en part laver individuelt arbejde udover det som laves fælles i gruppen, har denne part pligt til at formidle deres arbejde for de resterende parter.		
§5 - i tilfælde af konflikter i gruppen løses konflikten af en upartisk mægler.		
§6 - Gruppen skal være åben overfor alle slags spørgsmål og hjælpe med enhver tvivl, som en part måtte have.		
Misligholdes af gruppekontrakten medfører straf i form af bajere i BeerBox		
Denne kontrakt er udarbejdet af hele gruppen og alle grupper medlemmer har indvilgede i kontraktens paragraffer.		
Underskrift		
 Daniel Stuhr Petersen	 Simon Egebjerg Jensen	 Jakob Vingaard Petersen
 Rasmus Bak Petersen	 Jacob Lau Petersen	

Bilag 14 – Projektlog

Dato	Fremmødte	Hvad har vi nået i dag, og hvad er planen?
16-04-2018	Alle	ER, objektmodel, domænemodel Use cases for PBI 1 Tasks for database Statusmøde klokken 12
17-04-2018	Alle	GUI – lagerbeholdning og kodeimplementering Databaseforb. og stored procedure. 11:00 – møde med Jan Kl. 12 statusmøde
18-04-2018	Jacob, Jakob, Rasmus og Daniel	Begyndelse på rapport er i gang Refleksion over arbejdet indtil videre Domænemodellen skal kigges igennem igen Planlægning af møde med de andre grupper
19-04-2018	Alle	Plan for i dag: Møde med projektledelses-gruppen Møde med powerbanken.dk Påbegynder PBI nr. 2 Vi har besluttet at vi vil snakke om hvilke mål vi har for vores eksamensprojekt, hvad vi hver især gerne vil blive bedre til eller måske arbejde lidt mere med. (Tak projektledelses-gruppe) Vi har lavet SSD til CalculateOrderDate
20-04-2018	Alle	Plan for dagen: Testet CSV-reader, som følge af PO-møde Algoritme(DIF) Algoritme (Forventet salg) Algoritme (bestillings dato)
23-04-2018	Alle	Plan for dagen: Usecase for csvreader og algoritme forecast Implementering af ovenstående Test af forecast og csvreader Kvalitetskriterier af kode Planlægning af møde med underviser til imorgen

24-04-2018	Allesammen	<p>Plan for dagen:</p> <p>Kode: Algoritmer til beregning af ordredato mht. leadtime(fortsat) (er påbegyndt!)</p> <p>SD: CSVImport, beregning af ordredato</p> <p>DCD: relevante ændringer og opdateringer.</p> <p>Møde med Jan Brown</p> <p>Test til alortimen (test på view model) (mangler!)</p> <p>(GUI-implementering, wireframes, brugervenlighedstest). (skyder vi til senere!)</p>
25-04-2018	Minus Jakob	<p>Plan for dagen:</p> <p>Reflektere over kode og artefakter</p> <p>Lave nogle kommentarer til kode</p>
26-04-2018	Minus Jakob	<p>Plan for dagen:</p> <p>Køb hveder (Simon)</p> <p>Fix Database – SP, tabels, Design diagram (Rasmus)</p> <p>Rapport skrivning (forsættes) (Daniel, Simon og Jacob)</p> <p>Sprint Retrospektive</p> <p>(helst færdig inden kl. 12)</p> <p>Hvad der sket:</p> <p>Simon køber måske hveder?</p> <p>Databasen er fixet</p> <p>Der skrevet til 3 punkter i rapporten, herunder Scrum, kvalitetskriterier ved test, pakkediagram og lagdeling.</p> <p>Samt er der holdt retrospektive, hvor vi har snakket om hvad vi skal kigge på igen mandag, samt eventuelle ændringer i arbejdsprocessen.</p> <p>Sprint retrospective</p> <p>Vi har i dag snakket om hvad der har været godt, skidt og eventuelle ændringer til vores næste spring. Opsummering:</p> <p>Godt: projektlog, gode til at følge op, godt fremmøde, god struktur, artefakter og vurderingskriterier, gode til at arrangere møder, refaktivering.</p> <p>Skidt: folk arbejder på eget initiativ uden at forklare eller dokumentere, forhastede beslutninger, vi har ikke holdt os til PBI'erne.</p> <p>Ændringer: faste arbejdstider og pauser, unittests og TDD anvendes, (romkugle torsdag), plan for fordeling af arbejdsroller</p>
30-04-2018	Daniel, Rasmus og Simon	<p>Planlægning af sprint 2 (sprint planning)</p> <p>Rolleuddeling</p> <p>Nye tasks oprettes</p>

		Opdatering af scrumboard Scrum grooming PBI: Tilføj nye produkter er næsten færdiglavet.
01-05-2018	Rasmus, Jacob, Jakob og Daniel	Plan for dagen: Møde med Allan kl. 11:00. Sprint planning fortsat (fordi vi i dag er flere fremmødte) BPMN? Test?
02-05-2018	Allezusammen	Plan for dagen <ul style="list-style-type: none"> • Projektlog (skift scrummaster) • plan møde med projektgruppe • Baggrundsberregninger + refleksion • Grafik til kurver • Rapport skrivning (MVVM) • Wireframes <p>Vi har arbejdet på at lave ICommand og få løst vores problemer med MVVM.</p>
03-05-2018	alle undtagen Simon	Plan for dagen: <ul style="list-style-type: none"> • Møde med projektledere (13:00-13:30) • Refaktorering af lag • Rapport - tråde, ICommand, Udvælgelse af artefakter • Grafer (fortsat)
04-05-2018	alle undtagen Jacob	Plan for dagen: <ul style="list-style-type: none"> • eksperimenter • risiko-analyse • brugervenlighedstest • beregning af bestillingsdato
07-05-2018	Alle undtagen Simon	Plan for dagen: <ul style="list-style-type: none"> • Opdatering af Threads • Risici opdateres • Kig tilbage på sprint 2, og refaktorere wireframes og eksperimenter • Rapport refaktorering
08-05-2018	Simon, Rasmus, Jacob, Jakob og Daniel	Plan for dagen: <ul style="list-style-type: none"> • Ret PBI'er. • Key Performance Indicators tilføjes og rettes til. • Skrive et rapportafsnit om PBI'er. • Rapporten rettes for grammatiske fejl.

09-05-2018	Alle	<p>Plan for dagen:</p> <ul style="list-style-type: none"> • Retrospektive • Graf • Afslut sprint • Kig på PBI • <u>CODEWARS</u> <p><u>Retrospective:</u></p> <p>Godt:</p> <ul style="list-style-type: none"> • Kommunikation • God kageordning • Holder os til PBI's • Ingen forhastede beslutninger <p>Skidt:</p> <ul style="list-style-type: none"> • Dårlig afslutning - Ingen PO • Ikke så meget at lave • Har ikke fulgt helt op på sidste retrospective • Ikke fået lavet artefakter <p>Ændringer:</p> <ul style="list-style-type: none"> • Bedre kontakt til PO • Mere TDD!
14-05-2018	Alle undtagen Daniel	<p>Start på sprint 3!</p> <p>Planen for idag :</p> <ul style="list-style-type: none"> • Rapport renskrivning + handlingsplan • opdatering af DCD • KPI'er og burndownchart • Læse op på, hvad vi mangler af pensum i rapporten
15-05-2018	Alle	<ul style="list-style-type: none"> • Design af GUI • Forberedelse og planlægning af brugervenlighedstest. (fremgangsmåde, roller etc.) • Møde med Tove kl.11:00
16-05-2018	Allesammen er mødt på denne dag	<p>Plan for dagen:</p> <ul style="list-style-type: none"> • Kig og ret i GUI • Forberede brugervenlighedstest • Rapportskrivning • Møde med PO, med brugervenlighedstest samt få afsluttet sprintet.
17-05-2018		Experiment arbejde hjemme

18-05-2018	Alle	Plan for dagen: <ul style="list-style-type: none"> • Valuta attribut og aktiverings knap • Experiment rapportskrivning • Datoer i Datagrid
22-05-2018	Alle undtagen Simon	Plan for dagen: <ul style="list-style-type: none"> • Planlæg møde med Hans. • Rapportstruktur • ViewModels: der skal laves en viewmodel til hvert view. • Møde med Hans kl. 11:00.
23-05-2018	Alle undtagen Simon	Plan for dagen: <ul style="list-style-type: none"> • Beregning af datoer ændres <ul style="list-style-type: none"> • SD • Kode • osv. • Skrevet refleksion over burndown-chart. • Objekt- og domænemodel opdateres. (flueben) = true • Gennemlæsning af rapport.
24-05-2018	Alle fremmødte	Plan for dagen: <ul style="list-style-type: none"> • ret test • SD • bestemt periode i graf • Ret i kode
25-05-2018	Alle fremmødte	Plan for dagen: <ul style="list-style-type: none"> • Graf • Retrospektive • refakturering
28-05-2018	alle sammen	Plan for dagen: <ul style="list-style-type: none"> • Vi skal læse rapporten igennem • Samt udderigere task til at rette i rapporten
29-05-2018	Alle sammen	Plan for dagen: <ul style="list-style-type: none"> • Rapportskrivning • Tilføje afsnit til rapport • Gennemlæsning (fortsat)
30-05-2018	Alle undtagen Simon	Plan for dagen: <ul style="list-style-type: none"> • økonomiske og ikke-økonomiske gevinster: der tilføjes tekst! • Normalformer: der uddybes! • Sætte billede ind og skrive om statistikvinduet. • (hvis tid) kigger vi på systemet. • Litteraturliste laves. • Alle artefakter lægges i bilag på nær dem, som indgår i rapporten.