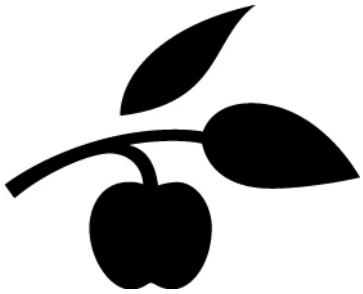


Syddansk Universitet

Teknisk Fakultet

Mærsk Mc-Kinney Møller Instituttet

Udvikling af Softwareprogrammer

Krediteringssoftware til TV2

Præsenteret af:		**Eksamens nr.**
Jonas Beltoft	jobel20@student.sdu.dk	493537
Hans Pedersen	haped20@student.sdu.dk	494042
Victor Bruun	vbruu20@student.sdu.dk	177620413
Jesper Diederichsen	jedie20@student.sdu.dk	496428
Casper M. Stillinge	casti19@student.sdu.dk	480325
Peter Ratgen	perat17@student.sdu.dk	454308

Vejleder:

Henrik Lykkegaard Larsen hlla@mmmi.sdu.dk

Semester: F21
Fagkode: T510043101
Gruppe: SE04
Aflevering: 31.08.2021

I Titelblad

Titel:	Udvikling af Softwaresystemer
Institution:	Syddansk Universitet, Det tekniske Fakultet Mærsk Mc-Kinney Møller Instituttet Campusvej 55, 5230 Odense M
Uddannelse:	Software Engineering
Semester:	2. Semester, F21
Kursuskode:	T510043101
Projektperiode:	F21
Omfang:	10 ECTS
Vejleder:	Henrik Lykkegaard Larsen
Projektgruppe:	SE04

I.1 Bidragserklæring

Ved at underskrive dette dokument bekræfter hvert enkelt gruppemedlem, at alle hæfter kollektivt for dokumentets indhold, samt at alle har bridraget til projektets udførelse.

Jonas Beltoft	`jobel20@student.sdu.dk`
Hans Pedersen	`haped20@student.sdu.dk`
Victor Bruun	`vbruu20@student.sdu.dk`
Jesper Diederichsen	`jedie20@student.sdu.dk`
Casper M. Stillinge	`casti19@student.sdu.dk`
Peter Ratgen	`perat17@student.sdu.dk`

II Resumé

III Forord

Denne rapport er udarbejdet af gruppe SE04 på software-engineering's bachelor studie gennem andet semester. Rapporten er skrevet med henblik på at oplyse læseren om, hvordan semesterprojektet er blevet udarbejdet, såvel som at vise hvordan det endelige slutprodukt fremstår. I denne rapport lægges der vægt på, at beskrive gruppens fremgangsmåde, tankeproces og arbejdsprocesser, for at kunne give et komplet billede af resultatet og vejen dertil. Gruppens arbejde med at programmere et krediteringssystem til TV2, som skal kunne erstatte de rulletekster, der normalt ville være efter endt program, har til formål, at kunne frigive mere tid mellem udsendelserne, som TV2 i stedet vil kunne bruge på ekstra programmer eller reklamer.

Dette semesters projekt omhandler brugen af databaser og optimeret objektorienteret programmering. Projektet er i forbindelse med vores to fag også bygget op omkring en udleveret case fra TV2. I denne case findes der en beskrivelse af hvilket system der skal udvikles, hvortil der er en række krav for, hvad systemet skal kunne. Løsningen af dette projekt er yderligere udvidet med flere implementeringer, som vi i gruppen vurderede nødvendige, for at slutproduktet er kategoriseret som færdig og optimalt.

Vores rapport er henvendt til enhver læser, som har en interesse inden for udvikling af et softwaresystem, samt processen som ligger bag udviklingen. Rapporten henvender sig også til de medarbejdere fra TV2 som har med vores case at gøre, for at give dem et indblik i udarbejdelsen af et softwaresystem, som kan supplere deres programmer i fremtiden.

Indholdsfortegnelse

IV Læsevejledning

Vi har som gruppe udarbejdet denne rapport over vores semesterprojekt, med hensigt på at skabe en overskuelig læseoplevelse. En læseoplevelse der skal beskrive vores tanker, beslutninger og de processer der danner rammer for vores projekt. Dertil har denne rapport til formål at belyse, diskutere og reflektere hvorvidt problemet for projektets case er løst og om vores produkt følger de givne krav.

Rapporten er bygget enkelt op, og for at skabe en let læseoplevelse anbefales det, at den læses fra toppen og ned. Derudover besidder rapporten et resumé som kan læses, hvis man ikke føler for at pløje igennem 40+ sider.

V Redaktionelt

Ansvarsområder			
Afsnit	**Ansvarlig**	**Bidrag fra**	**Kontrolleret af**
Forside	Victor Bruun		
Titleblad	Victor Bruun		
Resumé			
Forord	Victor Bruun		
Indholdsfortegnelse	Victor Bruun		
Læsevejledning	Victor Bruun		
Redaktionelt	Jesper Bork	Victor Bruun	
Indledning	Victor Bruun		
Fagligt vidensgrundlag	Alle		
Metode og planlægning			
Hovedtekst			
Krav	Jesper Bork		
Analyse			
Design			
Database Design	Jonas		
Implementering	Peter Ratgen Hans Petersen Jonas Beltoft		
Test	Peter Ratgen		
Diskussion			
Konklusion			
Perspektivering			
Procesevaluering			
Referenceliste			
Bilag			

1 Indledning

Dette semesters projekt tager udgangspunkt i TV2 og deres måde at håndtere krediteringer på. TV2 er en dansk tv-station, der blev grundlagt i 1988. TV2's forretning og den måde de tjener deres penge på, fungerer ved at lave kvalitetsbevidst tv til de danske stuer. TV2 gør deres tv-kanaler tilgængelige hos forskellige tv-udbydere, oveni at de viser reklamer mellem deres programmer, som skaber deres hovedsagelige indkomst.

TV2 er, ligesom alle andre tv-produktioner, en del af et stort marked, hvor tv-skærmen er deres vindue ud til forbrugeren. På sådanne et marked handler det om at få seeren til at bruge mere tid på sine egne kanaler, frem for konkurrentens. Hvis man vil holde styr på konkurrencen mellem kanalerne, vil det nemmeste nok være at kigge på, hvor mange seere der er på en given kanal, og hvor mange minutter seeren bruger på kanalen. Ud fra Kantars seereundersøgelser kan alle og enhver se seertallene på de forskellige tv-kanaler. Kantar indsamler nemlig seeretal hvert sekund, for at kunne levere de mest præcise seeretal [1] , som programlæggere i tv-branchen bruger til at forbedre tv-oplevelsen for seerene, og fastlægge hvilke tider i sendefladen, der er mest værdifulde. I følgende figur, som både er tilgængelig på Kantars hjemmeside, men også i vores case fra TV2 af, kan man se hvor mange minutter TV2s kanaler bliver set i forhold til de andre tv-kanaler.

Broadcaster	Antal min. dagligt	Antal min. ugentligt	Seerandel %
TV 2	74	519	51,5
DR	45	314	31,1
NENT	12	83	8,2
Discovery	6	43	4,2
Fox	3	19	1,9
Viacom	1	9	0,9
Disney	1	6	0,6
Turner	0	1	0,1
Andre kanaler	3	18	1,8

Figur 1: Seertid i uge 3 2020

Det første tal der ses i figur 1 viser, hvor mange minutter den gennemsnitlige dansker bruger på en given broadcasters tv-kanaler. Det næste tal viser hvor mange minutter det er om ugen, og det sidste tal viser, hvor stor en seerandel den givne broadcaster besidder på markedet for perioden. Som det ses har TV2 en stor seerandel, og af den grund består en vigtig del af deres indkomst af de reklamer de sender mellem deres programmer. Men den indkomst kan blive bedre endnu, hvis de bliver bedre til at optimere tiden mellem deres programmer, vurderer TV2. Og det er netop dette problem som danner grundlag for vores case.

1.1 Redegørelse for den udleverede case

Når et program slutter på en af TV2s kanaler, bliver der vist rulletekster for at kreditere de medvirkende i programmet. Disse rulletekster tager dog i nogle tilfælde op til 30 sekunder, hvilket er 30 sekunder for meget. TV2 har vurderet, at hvis man kan frigøre disse 30 sekunder fra rulletekster, og bruge dem på reklamer i stedet, vil de kunne tjene op imod 60 millioner DKK om året. [2]

Netop dette vil TV2 gøre ved at digitalisere krediteringerne, så de kan ses på nettet eller en app. For Udover at kunne sende flere reklamer mellem deres programmer, vil TV2 også have muligheden for at kreditere alle, og ikke bare de vigtigste efter endt program. I de fleste tilfælde var 30 sekunders rulletekster nemlig ikke nok til at kreditere alle medvirkende, så derfor har TV2 været nødt til at prioritere, hvilket selvfølgelig fører til at nogle krediteringer bliver glemt. Digitaliseres krediteringerne, vil alle kunne blive krediteret ligeligt, og der vil derved ikke være nogen som bliver glemt, fordi deres rolle er mindre vigtig.

Vores opgave i denne case er at skabe netop denne digitalisering af krediteringer. Det er vores opgave at udvikle et software system som kan tilføje, fjerne og redigere krediteringer i en database, hvortil vi har formuleret en problemformulering, som skal hjælpe os med at få udarbejdet en løsning på TV2s problematik med kreditering.

1.2 Formålet med opgaven

Formålet med det system vi har udviklet, er at gøre det lettere for producere, administratorer og TV2 som virksomhed, at kunne redigere og holde styr på krediteringer. Derudover er formålet med projektopgaven også at vi som gruppe skal forbedre vores evner inden for programmering af softwaresystemer og håndtering af databaser. Samtidigt med at vi videre udvikler vores kompetencer inden for gruppearbejde, og opsætning af et struktureret arbejdsmiljø.

Vores semesterprojekt er inddelt i to; først og fremmest har vi vores "inception", altså første halvdel. Denne del gik ud på at få udarbejdet et inceptionsdokument, som skulle lægge et fundament for vores videre arbejde med projektet. Formålet med dette inceptionsdokument var at vi som gruppe skulle ende med et veldefineret projektgrundlag, som indebar at der var styr på krav, scope og problemformuleringen. Udover dette skulle vi også klargøre, hvilke metoder vi ville benytte os af, for at kunne løse den problemformulering vi havde sat os for.

Som anden del af vores projekt, har vi den faktiske løsning af projektets problemformulering. Her har vi færdigudviklet koden, samt benyttet os af vores forarbejde i første del af projektet til at kunne besvare vores problem formulering så præcist som muligt.
Fordelen med at have delt vores projekt op i to halvdele, har været at vi som gruppe i første halvdel har kunne skabe os en forståelse for hvad problemet er, hvordan vi vil gribe casen an og hvordan vi som gruppe vil løse problemet på den bedst mulige måde. Dertil har vi så ef-

terfølgende, i anden del af projektet, kunne fokusere på at følge vores planer og benytte vores forarbejde, til at udvikle det bedst mulige softwaresystem, og besvare problem formuleringen uden problemer.

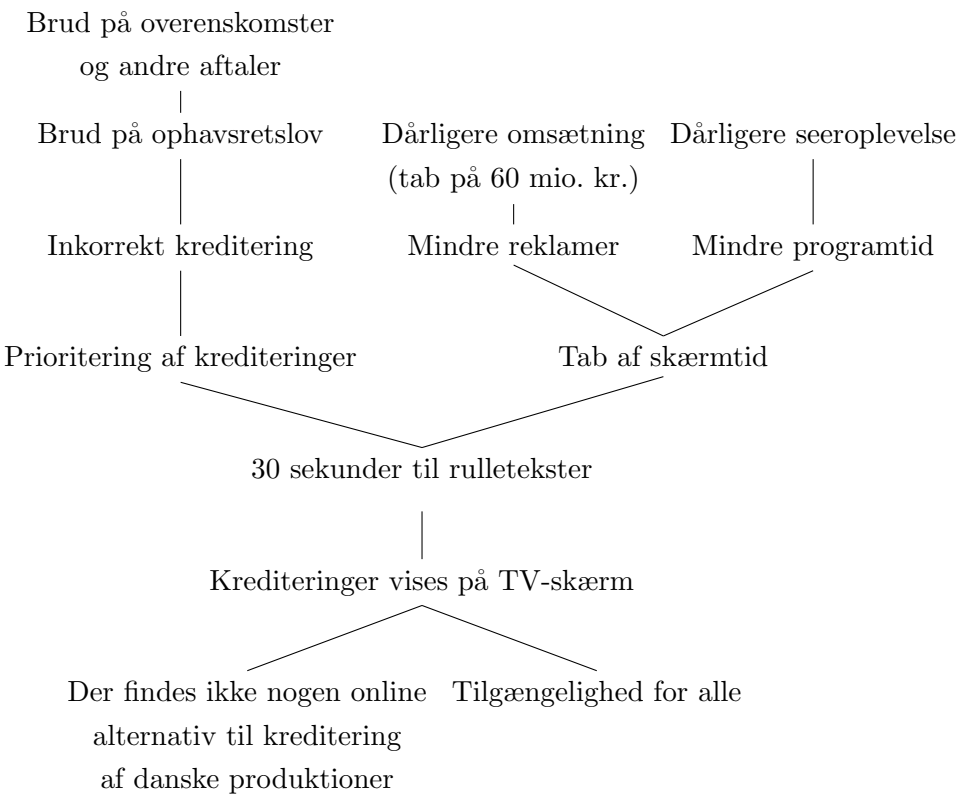
1.3 Problemanalyse

For at gå i dybden med den stillede case fra TV2, og dermed vurdere hvad det essentielle problem er, samt skabe os en problemformulering at arbejde ud fra, har vi først analyseret casen, med fokus på at identificere, hvilke mulige problemer der er. Den case som TV2 har lavet, er relativt udpenslet, og der er ikke meget tvivl om, at det essentielle problem for dem er, at det er forbesværligt at håndtere krediteringer til produktioner på den måde, som de gør i øjeblikket, nemlig at synliggøre krediteringer i 30 sekunder efter endt program i form af rulletekster. Denne nuværende løsning er et problem, da 30 sekunder ikke er nok til at kunne vise alle medvirkende i nogle udsendelser. [2] Der er også flere regler, som skal overholdes når der vises rulletekster, som f.eks:

- Ophavsretsloven
- TV 2s overenskomster
- Kontrakter for ophavsmænd og udøvende kunstnere
- Almindelig god skik

Ud fra de informationer vi har erhvervet os fra casen omkring TV2s regler for hvad og hvordan der skal angives krediteringer, har vi skabt et problem træ, der skal visualisere hvordan vi ser dette problem. Dette problemtræ ses i figur 2.

Kigger man på problemtræet, vil man kunne se, at de 30 sekunder, som der maksimalt må vises rulletekster i, skaber nogle problemer for TV2. TV2 mister nemlig skærmtid, af at skulle vise rulletekster efter hvert program, men udover det, har de, som tidligere nævnt, svært ved at få plads til samtlige krediteringer på 30 sekunder. Vi vurderer derfor at årsagen, bag disse problemer er, at TV2 er begrænset til at vise krediteringer i tv'et, og dette problem kan løses ved at flytte krediteringer over på et andet medium end tv-skærmen.



Figur 2: Problemtræ omkring de informationer, der er givet i casen.

1.4 Problemformulering

Ud fra vores analyse af TV2s case, er vi kommet frem til en problemformulering og dertilhørende underspørgsmål. Vores problemformulering har vi formuleret således:

Hvordan kan vi udvikle en prototype til et krediteringssystem, der vil kunne erstatte de klassiske rulletekster efter et afsluttet program?

Og vores underspørgsmål, til at supplere problemformuleringen lyder således:

- Hvordan er reglerne for krediteringer for danskproducerede programmer?
- Hvilke informationer skal og må inkluderes i krediteringer?
- Hvordan kan sådan et program gøres mest mulig brugervenligt(tilgængeligt) for både seer og administrator?
- Hvordan kan vi sikre et velstruktureret program, der mindsker fejl og ventetid samt strukturere databasen således, at der undgås fejl, duplikering af værdier og null-værdier?

1.5 Afgrænsninger

Projektet kan blive stort, og der er et utal af muligheder for, hvordan man kan skabe en overskuelig softwareløsnign på TV2s problem. Derfor har det også været vigtigt for os, at begrænse os. Vi vil gerne undgå at bruge for meget tid på, funktionaliteter, som ikke er vigtige, og lægge vores kræfter i de ting, som vi ved, vi kan overskue at gennemføre, og som der er tid til at få udviklet. Gruppen holder sig derfor til, udelukkende at arbejde med danske krediteringer, og undgår derved bevidst krediteringer til udenlandske shows og programmer. Gruppens produkt udarbejdes også kun som en prototype, og det betyder, at produktet ikke vil blive udviklet til et færdigt produkt, som vil opfylde alle givne krav fra TV2s side af. Krediteringerne vil dog stadig overholde de givne regler for krediteringer, men kommer til slut ikke til at være et produkt TV2 kan adaptere direkte over til.

2 Fagligt vidensgrundlag

Vidensgrundlaget for udarbejdelsen af denne rapport gennemgås i dette afsnit, med redegørelse for relevante begreber, teori samt det faglige vidensgrundlag. Dette afsnit tager udgangspunkt i det tilsvarende afsnit fra inceptionsdokumentet med tilføjelser samt udbedringer. Gruppens viden og færdigheder omkring udviklingen af software programmet, samt metode og planlægning, bygger på det lærte indhold fra de øvrige fag på 2. semester.

2.1 Kravudvikling

Når kravene til et system skal findes vil man som regel følge et workflow, der sikrer at kravene bliver så præcise, komplette og konsistente som muligt.

- Præcise krav, da tvetydige krav kan fortolkes på forskellige måder.
- Komplette krav, så de indeholder beskrivelser af alle aspekter.
- Konsistente krav der er konsekvente. Kravene må altså ikke modsige hinanden eller være i konflikt med hinanden.

Kravene til et softwaresystem er derfor vigtige at have styr på, da de fastsætter hvad systemet skal gøre, og definerer begrænsninger for dets udvikling og drift.

For at få et optimalt workflow til at få skabt nogle gode krav, har vi i gruppen kigget på at lave en brugsmønstermodel, også kaldet use-case. Det indebærer, at vi først har fundet de aktører, der forbindes med systemet. For at identificere en aktør kan man spørge: "Hvem eller hvad bruger systemet?Til hvad?"og "Hvorfor?". Man kan kigge på hvilken rolle de spiller i interaktionen med systemet. Man kan også spørge, hvilke andre systemer bruger eller bidrager med til systemet. Når man er afklaret med hvilke aktører der er, skriver man dem op i listeformat, hvor man inkluderer aktørernes navne, deres mål (hvad og hvorfor) og deres bidrag.

Næste punkt i en brugsmønster-model er at finde og identificere brugmønstre. Når man identificerer brugsmønstre vil man gå ud fra listen over aktører, og stille sig selv en række spørgsmål. Hvilke mål har en aktør? Det er skrevet ned i listen, og man kan derfor lave et brugsmønster til hvert mål aktøren har. Hvilke funktioner vil en specifik aktør have af systemet? Til det kan man lave et brugsmønster til en funktion, der giver målbare resultater til aktøren. Gemmer systemet information, som den henter frem? I så fald, for hvilken aktør gør den det? Dette er også værd at overveje, når vi skal lave brugsmønstrene. Ligeledes som med aktørerne, laver vi en liste over brugsmønstre, der indeholder deres navne og deres formål. Til slut skal der identificeres systemgrænser. Til det skal der først tegnes et brugsmønster diagram med aktører, de fundne brugmønstre, og linjer herimellem, der forbinder aktøren med de relevante brugsmønstre. Nu er det vigtigt, at man evaluerer, sine brugsmønstre, aktører og

emnet. Ud fra sine evalueringer kan man nu danne sig et billede af, hvilke krav, der stilles til systemet, og man ender derfor med en skitsering af en kravspecifikation til systemet.

2.2 Analyse af brugsmønstre

Brugsmønstrene kan vi derefter analysere og udarbejde sekvensdiagrammer og operationskontrakter, som vi kan overføre til et klassediagram og derved opbygge et overblik over systemet.

Dette foregår ved at der bliver udvalgt en række brugsmønstre til analysen. Til hvert af disse udformes et sekvens diagram, som viser det specifikke hændelsesforløb indefor et brugsmønster og de objekter og den adfærd der er beskrevet i brugsmønsteret.

Ud fra skevens diagrammet laves en operationskontrakt, som beskriver de forpligtigelser operationen har, samt hvilke ændringer der forekommer i systemet efter operationen er kaldt.

Dernæst kan et systemsekvens diagram udarbejdes. Et systemsekvens diagram er et udvidet sekvensdiagram, som viser den komplette række af begivenheder der udføres når operationen udføres. Til sidst kan det udvidet sekvens diagram overføres til et klassediagram.

2.3 Arkitektonisk design

Arkitekturen for systemet fastlægges tidligt i elaborationsfasen, når det er Unified Process der bliver benyttet. Det er vigtigt for ethvert software projekt at systemets arkitektur er velovervejet og godt udarbejdet, da det vil få stor betydning for det endelige produkt kvalitet. Designet fastlægges hovedsaligt ud fra de ikke-funktionelle krav, som f.eks. at vi i dette projekt har et krav om en tre-lags arkitektur, der allerede har sat bestemte rammer for det arkitektoniske design.

2.4 Brug af SCRUM i projektet

SCRUM er et agilt projektudviklings værktøj der primært er udviklet til udviklingen af software. SCRUM bygger overordnet på 3 artefakter. Product backlog, sprint og inkrementer.

Product backlog Product backlog er en liste af opgaver/krav der skal implementeres på et produkt for at nå et fælles produktmål. Emnerne i product backlog er prioriteret således at det mest nødvendige implementeres først. Disse emner er hvad der indgår i det næste SCRUM artefakt, sprintet.

Sprint Et sprint er en plan for at implementere en mængde af krav, for at opnå et mål. Man deler sprintplanlægningen op ved at besvare følgende spørgsmål om det. hvorfor, hvad og hvordan. Ved at finde ud af hvad målet for sprintet er, har man hvorfor sprintet er vigtigt. Derefter vælges de emner i product backlog, der, når implementeret, opnår dette mål. Disse emner sættes ind i et såkaldt sprint backlog der fungerer på samme måde som product backlog,

blot kun for et sprint. Dette sprint backlog er også prioritet er derved har man en plan for sprintet, og derved besvaret hvordan.

Inkrement Et inkrement er et betydeligt skridt på vej hen mod det endelige produkt. Et inkrement skal præsentere en virkende del af programmet. Det vil sige der hele vejen igennem er et kørende program, der udbygges med features inkrement efter inkrement.

SCRUM benytter sig også af nogle roller, der er essentielle for succesfuld brug af værktøjet.

SCRUM master En SCRUM master står får at få etableret SCRUM-flowet. Dette gøres ved at de skaber en fælles forståelse for hvordan arbejdsprocessen ser ud, og hvilket værktøjer der anvendes og hvordan. Det er SCRUM masterens opgave at sørge for hele processen bliver overhold, for hvis ikke dette formås, forsvinder alle fordelene ved SCRUM. I Gruppen har Hans Pedersen ageret som SCRUM master

Product owner En product owner er ansvarlig for product backlog. Det er vigtigt at de emner der skal op på product backlog, går gennem product owner, så de har det fulde overblik over hvad produktet indeholder. Derfor er der også kun én product owner, så man sikrer sig at der altid er en der har 100% styr på hvordan produktet ser ud, og hvordan det skal se ud. Idet gruppen er lille nok til kun at indeholde et SCRUM team, har gruppen udnævnt at product master er hovedansvarlig for produktet. I gruppen har Jonas Beltoft ageret som product owner, da han er iderig, og har gode evner indenfor programmering, og visualisere sig programmer.

Developers Developers, eller udviklere, er dem der sørger for at et sprint rent faktisk også arbejder fremad mod et inkrement. De står for at implementere de emner der er tilføjet til sprint backlog I gruppen er alle medlemmer indgået som udviklere.

2.5 Objekt-orienteret programmering

Hvad er OOP? Objekt-orienteret programmering (eller OOP), er et programmeringsparadigme baseret på et koncept af "objekter". Disse objekter kan indeholde en række data, generelt kaldet attributter. Kode kan være skrevet i form af en procedure, der skal lave noget bestemt, dette bliver kaldt en metode. Et eksempel kunne være at kode en person, denne person ville være et objekt. Personen har flere forskellige metoder til at kunne tale, bevæge sig, trække vejret OSV. Person objektet skal også have nogle attributter så som en højde, deres køn, deres alder osv.

Ved brug af Java-kodesprog i dette projekt kommer vi til at gøre stor brug af klasser. Når man definerer en klasse, opretter man egentlig et blueprint til et objekt. Heri defineres ikke direkte data, men der defineres hvad der skal gøres med noget data der er givet til objektet.

2.5.1 OOP's fire basale koncepter

Indkapsling Når man snakker om indkapsling inden for OOP, så betyder det at nogle data eller noget funktionalitet bliver indkapslet. Dette gøres ved hjælp af Access modifiers. Access modifiers hjælper til at bestemme, hvem der har adgang til data og funktioner, og hvor meget der er adgang til. Der findes en række forskellige access modifiers som kan ses på tabellen 1 på side 9. Indkapsling er en god måde at lave dataskjulning for andre klasser. Indkapsling gør det også nemmere at teste produktets sikkerhed.

Access Modifier	**Inden for klassen**	**Inden for pakken**	**Uden for pakken kun af subklasser**	**Uden for pakken**
Private	X			
Default	X	X		
Protected	X	X	X	
Public	X	X	X	X

Tabel 1: Liste over tilgange ved de forskellige access modifiers

Abstraktion Abstraktion er en måde på at gemme specifikke værdier og metoder. Abstraktion kan ses som en udvidelse på indkapsling, siden det kan gemme information eller metoder for ekstern kode. Dette gør interfacet af objekter meget simpelt. F.eks. kan abstraktion forstås ved at kigge på kroppen, vores skind fungere som an abstraktion til at gemme hvad der foregår inde i kroppen.

Nedarvning Nedarvning kommer fra at være en subklasse. Her nedarver subklassen attributter og metoder fra dens superklasse. Dette gøre det muligt at nemmere kunne genbruge kodestykker. Derudover bruges det også til metodeoverskrivning så man kan opnå runtime polymorfi. Imodsætning til nedarvning, hvilket er it såkaldt "IS-A"forhold, findes der også et "HAS-A"forhold, et aggregat. Et HAS-A forhold beskriver de gange hvor et objekt gøre brug af et andet objekt til at gemme data. F.eks. hvis man har en kunde med nogle informationer inde i et objekt, så kan der herunder oprettes et adresse objekt hvilket indeholder information som by, kommune, post nummer osv. Et aggregat kan være god at bruge i situationer hvor en stor klasses information kan deles ud i mindre bidder. Det hjælper også meget på kode genanvendelse. I den virkelige verden kan vi forestille os nedarvning ved at kigge på insekter. Alle insekter har lignende egenskaber så som at have seks ben eller et exoskelet. Altså ville en grasshoppe eller myre have nedarvet disse egenskaber fra superklassen af insekter.

Polymorfi Polymorfi er muligheden for at et objekt kan tage flere former. Dette gør at vi kan implementere en metode fra en superklasse i en subklasse på forskellige måder. Det ville

altså sige at hvilket som helst subklasse objekt i Java kan hvilken som helst form som er i superklassens hierarki, inkluderende sig selv. Generelt er der to typer af polymorfi:

- Dynamisk polymorfi

  Dynamisk polymorfi er også kendt som run-time polymorfi. Dette kan beskrives ved at tænke på superklasse-subklasse forhold. De begge har den samme metode, og subklassen har overskrevet superklassens metode (metode overskrivelse). Når et objekt bliver tildelt en klasse reference, og en metode fra objektet bliver kaldt, så er det metoden indeni objektets klasse, der bliver udført. Der bliver altså ikke kaldt metoden fra reference klassen, hvis altså referencen er en superklasse.

- Statisk polymorfi

  Statisk polymorfi er også kendt som compile-time polymorfi eller metode overbelastning. Dette betyder, at man har angivet flere af de samme metodenavne, men med forskellige argumenter. Det betyder altså, når man kalder sådan en metode, så ville kompilatoren beslutte hvilken metode, der bliver kørt i forhold til hvilke parametre, der er angivet.

2.6 Trelags arkitektur

Trelags arkitekturen er en software arkitektur, der deler programmet op i tre lag. Et præsentationslag, et logiklag og et datalag:

- Et præsentationslag som udelukkende står for at forsyne brugeren med en brugerflade og hente information fra brugeren og med den information kommunikere med det næste lag, logiklaget.

- Logiklaget står for alle de logiske operationer som er associeret med de gældende elementer i præsentationslaget på baggrund af den information, som brugeren har indtastet. Logiklaget kommunikerer også med datalaget, og kan her sammenholde information fra brugeren med det i databasen for at tage en beslutning om, hvad programmet skal gøre.

- Datalaget står for at lagre systemets data som logiklaget kan anvende.

For en succesfuld anvendelse af trelags arkitekturen vil brugeren udelukkende blive præsenteret for præsentationslaget og behøver ingen viden om de underliggende lag, da al kommunikation mellem bruger og program sker i dette lag. Brugen af trelags arkitekturen giver nogle fordele.

- Først og fremmest kan udarbejdelsen af de forskellige lag uddelegeres således at de udvikles samtidig. Dette gør udviklingen af programmet hurtigere

- Da al logikken sker et sted, bliver programmet mere skalérbart idet at koden/programmet ikke er kludret sammen og har for mange indgangspunkter.

- Hvis et lag bryder sammen eller ikke virker, påvirkes de andre ikke, da de er opdelt.
- Grundet at trelags arkitekturen opdeler præsentationslaget og datalaget med et logiklag, vil man med korrekt implementering kunne sikre, at en bruger ikke får tilladelse til at tilgå data, som ikke er tiltænkt dem.

2.7 Unified Process

Unified Process (UP) er en iterativ og trinvis udviklingsproces. UP er delt op i 4 trin, eller faser, disse og deres definitioner er:

- Inceptionsfasen

  I inceptionsfasen skal man finde ud af projektets gennemførlighed, finde frem til vigtige krav og identificere potentielle risici.

- Elaborationsfasen

  I elaborationsfasen skal der laves en arkitekturprototype, oveni at risikovurderingen forbedres. Hertil skal hovedparten af brugsmønstrene findes, samt at der skal lægges en plan for konstruktionsfasens forløb.

- Konstruktionsfasen

  I konstruktionsfasen færdiggøres den endelige identifikation af brugsmønstre, samt disses beskrivelse og realisering. Analyse, design, implementation og test færdiggøres også i denne fase. Undervejs redigeres projektets risikovurdering.

- Transitionsfasen

  I denne fase rettes fejl, og brugerne forberedes på at anvende softwaren. Her laves også manualerne og anden dokumentation, til sidst gennemføres et review af projektet.

En vigtig pointe omkring UP er at det er en proces drevet af brugsmønstre (use cases), samt at der fokuseres på risikostyring og arkitektur. Og som tidligere nævnt er der fokus på iterationer og en trinvis udviklingsprocess. Dette betyder at systemet udvikler sig trinvist for hver iteration.

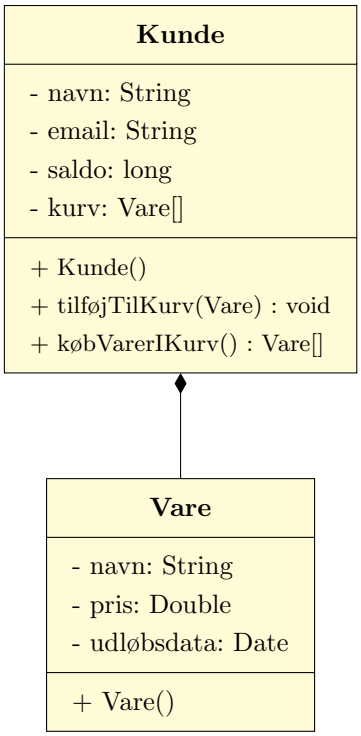
2.8 UML

UML eller Unified Modeling Language, tilbyder en måde at visualisere et systems arkitekturelle blueprints i et diagram.

2.8.1 Klassediagrammer

Det er bl.a. brugt til at lave Class Diagrams, da det giver en oversigt over hvilke klasser der er i et program, hvilke attributter og metoder de har, samt disses synlighed (Access Modifiers). UML anfører også sammenhængen mellem klasserne, om de fx nedarver fra klassen eller andet, hvilket er en stor del af grunden til at man bruger UML. Ud fra det kan man aflæse hvordan forskellige klasser bliver brugt andre steder i programmet, og derved få en bedre forståelse for hvordan programmet virker, uden at man har set programmet køre. Der er særlig notation til at fremhæve alle OOP's fire basale koncepter,

hvilket gør det muligt at udarbejde og/eller visualisere præcist det program, der er ønsket. Øverst ses navnet på den klasse, der beskrives. I sektionen under, er alle klassens attributter, og under dem er klassens metoder. Hvis klassen har en eller flere constructors, kan disse også ses, da de har det samme navn som klassen. På grund af alt dette, kan UML være et meget kraftfuldt værktøj under udviklingen af et objektorienteret program, da man kan visualisere alle de objekter og klasser, der bliver lavet og instantieret. Samt sammenhængen mellem disse objekter og klasser. UML bruges også til andre ting, såsom brugsmønster-diagrammer, som giver et overblik over aktører og de brugsmønstrer de interagerer med. Dette er en anderledes måde at modellere på, men det er alt sammen under UML, da det er en universel måde at forbinde elementer på, i en visuel kontekst.



Figur 3: UML klassediagram for sammenhængen mellem en Kunde og en Vare

2.8.2 Beskrivelse af database med UML

UML kan også bruges til at vise en database. Det ses nemt hvilken udgave, der er brugt, da man her vil se notationer så som {PK} som beskriver at denne attribut er en primær nøgle i denne tabel. Hver kasse er her ikke en klasse, men en tabel. Den har kun attributter, da en tabel ikke har funktioner. Et UML diagram over en database vil også have forskellige måder at visualisere sammenhæng mellem tabellerne. Klassediagrammet som beskrevet har kun én standardiseret måde at forbinde klasserne på, hvilket gør det nemmere at lære og hurtigere at finde ud af, hvad der menes med de forskellige symboler man støder på, hvorimod et UML diagram over en database har flere forskellige måder at vise sammenhæng på, selvom de viser det samme. I klassediagrammet vil man se symboler som f.eks. betyder at denne klasse nedarver fra en anden klasse, eller bl.a. viser at den ene klasse "bruger"eller "afhænger"af en anden klasse. Dette finder man ikke i et data-

base UML diagram. Her viser forbindelserne, hvor mange elementer, der kan være, f.eks. kan en kunde købe mange varer, men en type vare kan købes af mange forskellige kunder, så det vil kaldes en mange til mange forbindelse. Dette kan noteres på forskellige måder, f.eks. med crows foot, Chen eller andre. Vi som gruppe vil bruge Chen, som er en meget simpel måde at vise, om det er 0, 1 eller flere. Disse notationsformer er også brugt i ER og EER modeller, men det betyder ikke at et UML diagram er det samme som et ER eller EER diagram, da de ikke går direkte ned i databasen, men er mere en overordnet beskrivelse og kan vise andre ting end UML kan.

2.9 MoSCoW

MoSCoW er en model der bruges når en liste af krav skal prioriteres. Kravene indeles i 4 grupper, efter hvor vigtige de er for projektets succes. Her startes der med "Must Have"som er de vigtigste krav og som er nødvendige for at projekt bliver en succes. Dernæst kommer "Should Have"som er krav, der ikke er nødvendige, men det vurderes at de giver en tilstrækkelig værdi til projektet, i forhold til arbejdsindsatsen. "Could Have"er krav som ikke er vigtige for projekts gennemførsel, de kan tages med i tilfælde af overskydende arbejdsressourcer, men der planlægges ikke efter, at de skal medtages. Til sidst findes "Would Have"som er krav, der giver mening for projektet, men er valgt fra i den pågældende udgave af projektet.

2.10 FURPS+

- Usability

  "Usability"handler om de menneskelige interaktioner med programmet. Det gælder om at se på, hvor effektivt programmet er. Er dokumentationen for programmet i orden og uddybende nok, til at forklare hvad programmet står for.

- Reliability

  "Reliability"omhandler det, der vedrører oppetid, præcision i systemets beregninger.

- Performance

  "Performance"drejer sig om, hvordan systemet skal reagere på større mængder af information.

- Supportability

  "Supportability"omhandler krav, som vedrører testbarhed, kompatibilitet, konfigurerbarhed.

- Plus

  "Plus"er den del af FURPS+ der indeholder begrænsninger og specifikke krav for systemet.

- Design constraints

  Man kigger på designmæssige begrænsninger, der kunne være for projektet. Ændrer I/O enheder eller den valgte DBMS, hvordan softwaren skal bygges?

- Implementation requirements

  Man kigger også på implementationen og dets krav, og beskriver, hvordan programmørene skal forholde sig. Skal de bare forholde sig til standarderne?

- Interface requirements

  Interface krav kigger på om, der er nogle interfaces som programmet skal fungere på eller med.

- Hardware requirements

  Til sidst er der hardware krav, hvor der kigges på, om der er nogle krav til hvilken hardware systemet skal interagere med eller kører på, og om det skaber nogle forudsætninger for produktionen.

3 Metoder og planlægning

I løbet af projektet har gruppen benyttet både Unified Process (UP) og Scrum. Overordnet set har gruppen fulgt UPs faser og iterationer fra inceptionsfasen til transitionsfasen, dette har givet struktur og målrettet det forberedende arbejde op til elaborationsfasen, hvor det faktiske udviklingsarbejde begyndte. Det var i denne elaborationsfase vi gjorde brug af Scrums agile projektudviklings værktøj. Der vil ikke blive yderligere redegjort for arbejdet i inceptionsfasen, da det alt sammen er dokumenteret i inceptionsdokumentet i bilag G.
Brugen af Scrum i elaborationsfasen og tilmed konstruktionsfasen, har fungeret godt for gruppen. Dette skyldes at hvert Scrum sprint har givet et "releaseable"produkt, lige vel som hver iteration i UP kræver. Ydermere har Scrums sprint metode været, en god hjælp til at fastlæge deadlines, for enkelte dele af udviklingsarbejdet.

3.1 Plan i elaborationsfasen

4 Hovedtekst

I dette afsnit dokumenteres det faktiske arbejde, samt de opnåede resultater fra både 1. og 2. iteration.

4.1 Overordnet krav

De overordnede krav er udformet på baggrund af den stillede case fra TV 2, samt et opfølgende interview med Morten Lehm, som er udvikler hos TV 2. De overordnede krav blev sat under

arbejdet i inceptionsfasen, men er sidenhen tilpasset og revideret, derfor indholder dette afsnit en opdaterede udformning af de overordnede krav, samt prioritering. Disse funktionelle krav afspejler brugmønstrende, hvilket de derfor skal forstås som brugmønstre lige vel som krav.

ID	**Funktionelle Krav**	**Beskrivelse**	**MoSCoW**
F/B 01	Registrering af kreditering	Det skal være muligt for producere og firmaer at tilføje programmer og kreditere medvirkende hertil.	Must
F/B 02	Bruger system	Prototypen af forbrugersystemet skal gøre det muligt for forbrugeren at få informationer om krediteringer og produktioner.	Must
F/B 03	Søgning på kreditering	Det skal være muligt at slå en person op og derved se samtlige programmer en person har været med i.	Must
F/B 04	Oprettelse af personer og programmer	Der skal implementeres en skabelon, som skal udfyldes, når der skal oprettes en person eller et program.	Must
F/B 05	Rediger krediteringer	Det skal være muligt at redigere i eksisterende krediteringer.	Must
F/B 06	Anvendelse af data	Det bør være muligt for TV 2 at eksportere en specifik mængde af data i forskellige formater som XML og CSV, så TV 2 nemt kan anvende dette data på anden vis.	Should
F/B 07	Nem tilgang til kreditering	Efter endt show vises en kode på skærmen, som kan slås op i programmet og krediteringene til det pågældende program vises	Should
F/B 08	Mulighed for at skelne mellem personer og firmaer	Hver person og firma har sit eget unikke ID, hvis det er for svært at skelne på ID, kan der forsøges at hente yderligere oplysninger som f.eks. telefonnummer, kaldenavn, alder.	Should
F/B 09	Fleksibel søgning	Der behøves ikke vælges, hvad en bruger søger efter, men programmet kan matche inputtet med alle mulige entries (uanset om det er program eller person). Brugeren skal også have mulighed for eksplicit at vælge hvilken type (person, program osv.)	Should

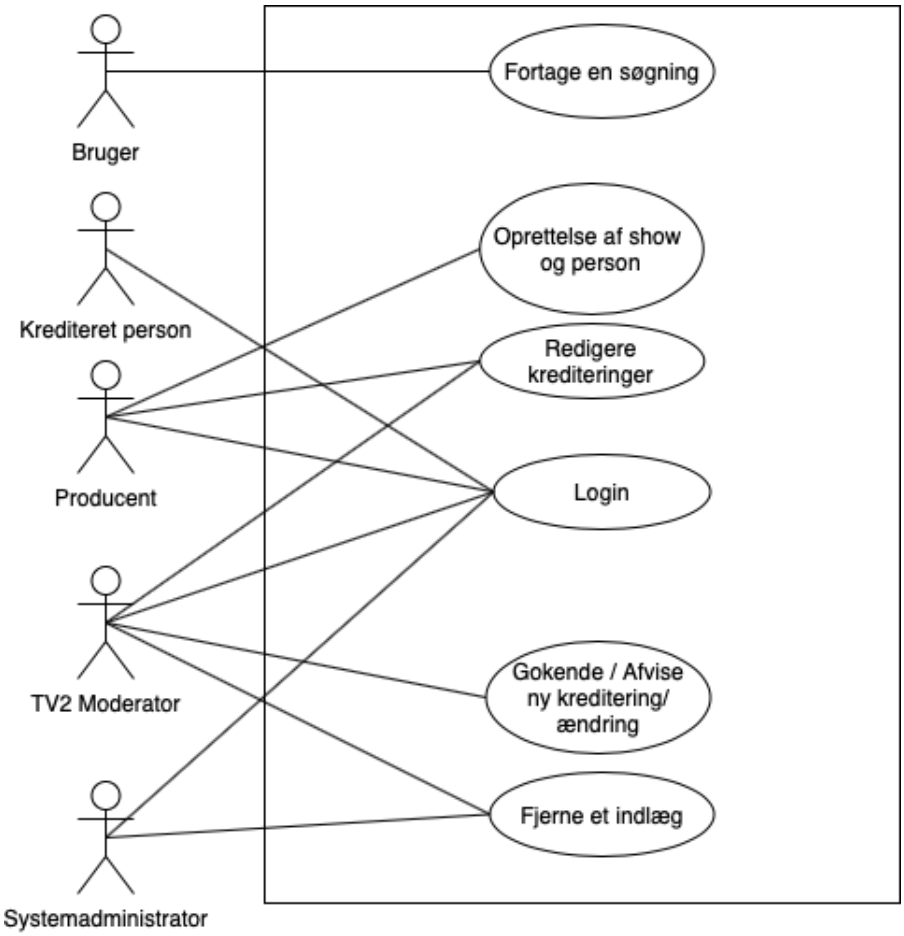
F/B 10	Adgangskontrol	Der skal implementeres en form for adgangskontrol, for hvem der skal have adgang til redigering, oprette og slette data.	Should
F/B 11	Rapportering	Det bør være muligt for TV 2 at eksportere en specifik mængde af data i forskellige formater som XML og CSV, så TV 2 nemt kan anvende dette data på anden vis.	Could
F/B 12	Valg af sprog	Det bør være muligt at skifte mellem sprog, som minimum dansk og engelsk.	Could
F/B 13	Notifikationer	Hver gang der sker noget i databasen skal TV 2's medarbejder modtage en notifikation med ændringer.	Would

Tabel 2: Funktionelle krav - Revideret prioriteringer i MoSCoW

Ovenfor i tabel 2 ses den endelige prioritering af krav. Siden inceptionsfasen er "Adgangskontrol"blevet nedprioriteret, da arbejdet med andre funktioner synes mere vigtig og giver større værdi for produktet, dog er en alternativ login funktion implementeret for at skelne mellem aktørenes rettigheder. Derudover er kravet om redigering tilføjet som et "Must"krav.

4.1.1 Brugsmønster diagram

Ud fra kravene/brugsmønstrende opstilles et brugsmønster diagram, som viser interaktionenerne mellem aktører og brugsmønstre. Da det ikke er alle krav/brugsmønstre, som er implementeret i produktet, ses nedenfor et brugsmønster diagram (Figur 4) over de opnåede funktioner.



Figur 4: Opdateret brugsmønster diagram

Ovenfor ses et opdateret brugsmønster diagram i Figur 4, med en række aktører. Aktørene er her opstillet i hierarkisk rækkefølge fra den almene bruger til systemadministratoren, dvs. at alle aktører har de samme rettigheder som den overstående. I dette diagram findes enkelte "overflødige"aktører, dette skyldes at der er tiltænkt bestemte rettigheder til disse aktører, som er blevet nedprioriteret og ikke er implementeret i denne udgave. Disse overflødige aktører er listet op i diagrammet da de stadig er en del af de alternativ login system, og derfor stadig eksistere i programmet.

4.1.2 Supplerende krav

Ud over de funktionelle krav og brugsmøsntrende har gruppen udarbejdet end række supplerende krav, som er med til at udforme de ikke-funktionelle krav. Disse krav er udformet på baggrund af casen stillet af TV 2, og enkelte krav til design, som er stillet af SDU.

Opnået	
3-lags arkitektur	Det ønskes at programmet følger en 3-lags arkitektur.
Oppetid	TV 2 har et krav om at systemet har en oppetid på 99,7 - 99,8% .
Svartid	TV 2 Ønsker at svartiden på systemet skal være så lav som muligt.
Delvist opnået	
Sikkerhed	TV 2 stiller et meget stærkt krav om, at der er styr på, at al data i systemet er korrekt, da data skal anvendes for at sikre, at krediterede personer får den betaling, de har ret til.
Unikke personer	Der er et krav om, at man skal kunne differentiere personer med f.eks. identiske navne.
Ikke opnået	
Integration med andre systemer	TV 2 har et ønske om at programmet skal kunne integreres med en række af de systemer, de anvender i forvejen. Eksempler kunne være EPG og login-systemet til TV 2 Play.

Tabel 3: Supplerende krav

I Tabel 3 overfor er det supplerende krav, som blev udformet tilbage i inceptionsfasen. Dog er ikke alle krav er opnået i den forventede udstrækning. Kravet om sikkerhed er placeret under delvist opnået, da funktionen om at krediteringer indtastet af producenter, skal godkendes af en TV 2 moderater er implementeret. Dog er et aktiv login system ikke implementeret, og derfor vurderes det at kravet ikke er opfyldt fuldt ud, øvrige supplerende krav er uddybet i afsnit 4.2.1.

4.2 Detaljeret krav

Ud fra de overordnede krav er enkelte krav/brugsmøsntre udvalgt til en mere detaljeret beskrivelse. Brugsmønstrene er udvalgt på baggrund af vigtigheden for projektet, derfor er det brugsmøsntrene "registrering af kreditering"og "søgning på kreditering", der er medtaget i rapporten.

Brugsmønster: Registrering af kreditering.
ID: B01
Primære aktører: Producenter og firmaer (Opretter)
Sekundære aktører:
Kort beskrivelse: Producent for showet eller en SoMe-ansvarlig fra et firma kan logge ind på systemet og oprette showet, ved at udfylde en skabelon.
Præconditioner Man skal være logget ind som enten producent eller firma

Hovedhændelsesforløb 1. Producent eller firma logger ind på systemet. 2. Opretteren navigerer til siden med oprettelse af show. 3. Opretteren udfylder skabelon til oprettelse af show som indbærer: • Navn på show og evt. serie ID. • Dato for første udsendelse. • Navn på producent. • kategori. • mm. 4. Krediteret personer tilføjes til show. • Opretteren skal kunne hente personer, som eksisterer i databasen og tilføje dem til showet. • Ved tilføjelse af ikke eksisterende person: se alternativt hændelsesforløb. 5. Den udfyldte skabelon registreres og sendes til databasen som "ikke godkendt", hvorefter en TV 2 moderator kan tilgå krediteringen og godkende den.
Postkonditioner: Showet står som "ikke godkendt"indtil en ansat fra TV 2 har kvalitetstjekket indholdet og godkendt det.
Alternative hændelsesforløb: Skridt 4. Ved tilføjelse af personer, der ikke allerede eksisterer i databasen, får opretteren mulighed for tilføjelse af den nye person ved et pop-up vindue, for yderligere specificering se B02 - Oprettelse af person.

Tabel 4: Detaljeret brugsmønster over B03

Brugsmønster: Søgning på kreditering
ID: B03
Primære aktører: Bruger
Sekundære aktører:
Kort beskrivelse: Brugeren bliver mødt af et søgefelt, hvor de indtaster deres ønskede søgekriterier og vælger herefter hvilken kreditering, de ønsker at få vist
Præconditioner: Brugeren har et søgekriterie i form af navn på show/person eller ID

Hovedhændelsesforløb: 1. Brugeren bliver mødt af et tomt søgefelt 2. Brugeren indtaster et søgekriterie i form af navn på show/person eller ID 3. Brugeren søger med de givne søgekriterier 4. Brugeren får vist en liste af krediteringer, der matcher søgekriterierne 5. Brugeren vælger her den ønskede kreditering 6. Brugeren får nu vist den specifikke kreditering som de har valgt
Postkonditioner: Logiklaget har leveret informationer til brugeren, og brugeren har adgang til de krediteringsinformationer de ønsker
Alternative hændelsesforløb: • Skridt 5: Hvis brugeren ikke kan finde den ønskede kreditering kan de fortage endnu en søgning eller opgive

Tabel 5: Detaljeret brugsmønster over B01

4.2.1 Supplerende krav

En detaljeret beskrivelse af de supplerende krav kan med fordel stilles op ved hjælp Furps+ model. Med Furps+ kategoriseres kravene efter hvilke område de vedrører.

- **Reliability**
  Under dette punkt placeres de krav, som bliver sat til pålideligheden af programmet. I vores tilfælde har vi 2 krav, som omhandler oppetid og sikkerhed for programmet.

Oppetid	TV 2 har et krav om at systemet har en oppetid på 99,7 - 99,8% .
Sikkerhed	TV 2 stiller et meget stærkt krav om, at der er styr på, at al data i systemet er korrekt, da data skal anvendes for at sikre, at krediterede personer får den betaling, de har ret til.

Tabel 6: Supplerende krav - Reliability

- **Performance**

Til dette punkt har vi et enkelt krav, som ikke er yderligere specificeret end, at programmet skal have en svartid der er så lav som muligt.

Svartid	TV 2 Ønsker at svartiden på systemet skal være så lav som muligt.

Tabel 7: Supplerende krav - Performance

- **Plus**

  Design constraints: Her er det vi placere de specifikke begrænsning vores system er pålagt. Kravet om 3-lags arkitektur placeres her da det er et specifikt krav til systemet stillet af SDU.

3-lags arkitektur	Det ønskes at programmet følger en 3-lags arkitektur.

Tabel 8: Supplerende krav - Design constraints

Implementation requirements: Implementerings krav stiller retningslinjer for programmørenes implementering, og de blot skal forholde sig til standarderne eller noget specifikt. Her har vi et krav stillet TV 2, som omhandler måden hvorpå en person registreres i systemet. Det skal her være muligt at differentiere mellem personer med identiske navne. Dette vurdere vi til at være delvist opnået, da det er muligt at oprette personer med identiske navne, og de vil få tildelt unikke ID'er. Dette er dog ikke tydliggjordt for brugeren, hvorfor vi mener den er delvist opnået.

Unikke personer	Der er et krav om, at man skal kunne differentiere personer med f.eks. identiske navne.

Tabel 9: Supplerende krav - Implementation requirements

Interface requirements: Interface requrements kigger på om der nogle interfaces, som programmet skal kunne på eller med. Til dette projekt har TV 2 haft et ønske om at programmet skal interagere med deres øvrige programmer, dette har dog ikke været muligt at udfører i denne udgave af produktet.

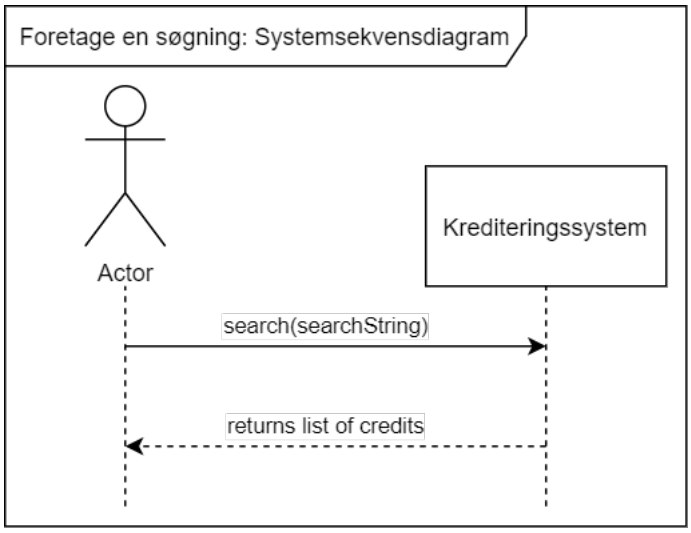
Integration med andre systemer	TV 2 har et ønske om at programmet skal kunne integreres med en række af de systemer, de anvender i forvejen. Eksempler kunne være EPG og login-systemet til TV 2 Play.

Tabel 10: Supplerende krav - Interface requirements

4.3 Analyse

4.3.1 Brugsmønsterrealisering

B03: Foretage en søgning Først foretages en brugsmønsterrealisering på brugsmønsteret B03 foretage en søgning. Først dannes der et systemsekvensdiagram. Dette giver et overblik over hvad målet er. Systemsekvensdiagrammet for B03 ses på firgur 5



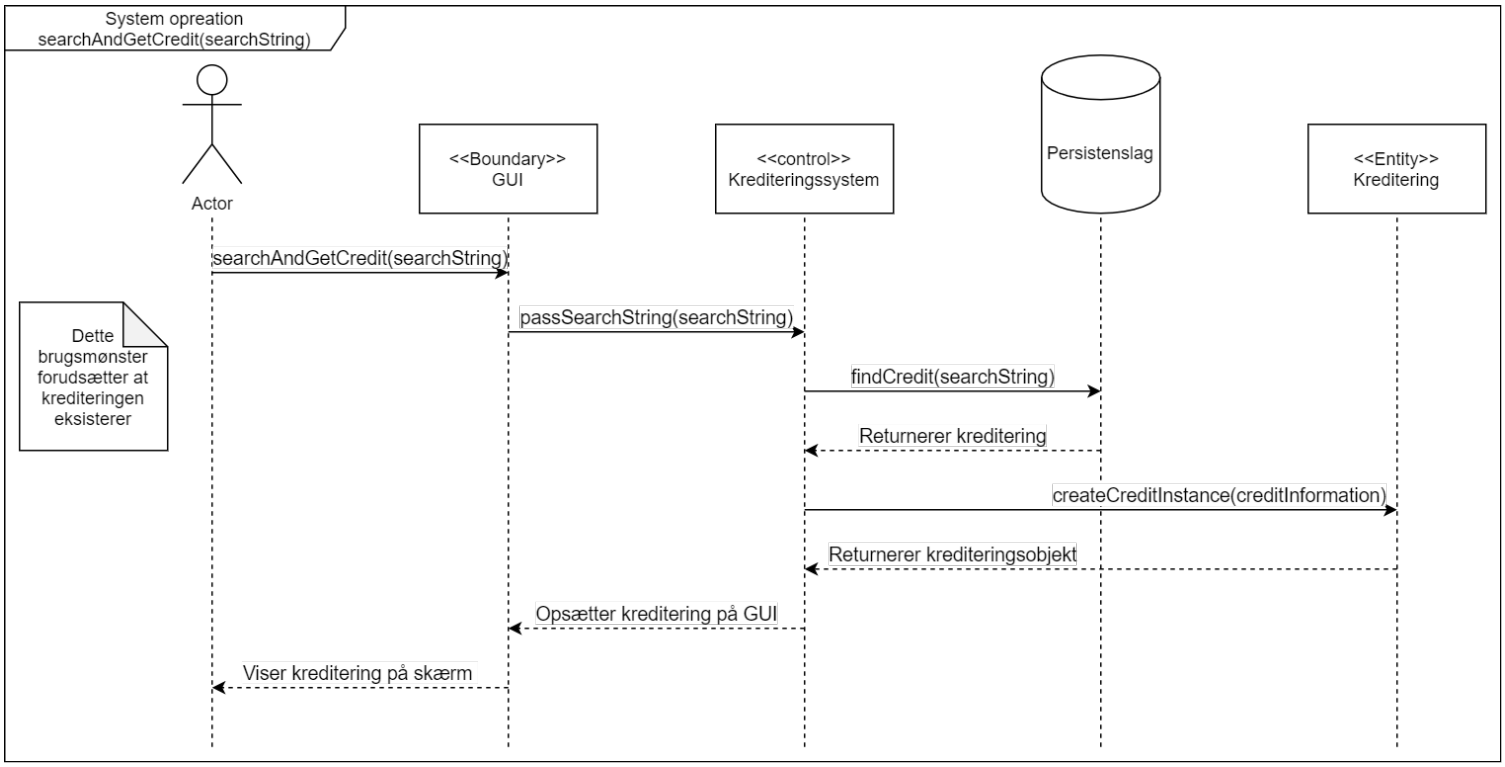
Figur 5: Systemsekvensdiagram for search

Firgur 5 viser at en aktør, her en bruger, bruger funktionen search på krediteringssystem som derefter returnerer en liste af krediteringer. Ud fra dette diagram blev der opsat en operationskontrakt over hvilket ansvar search operationen har. Denne operationskontrakt ses på tabel 11

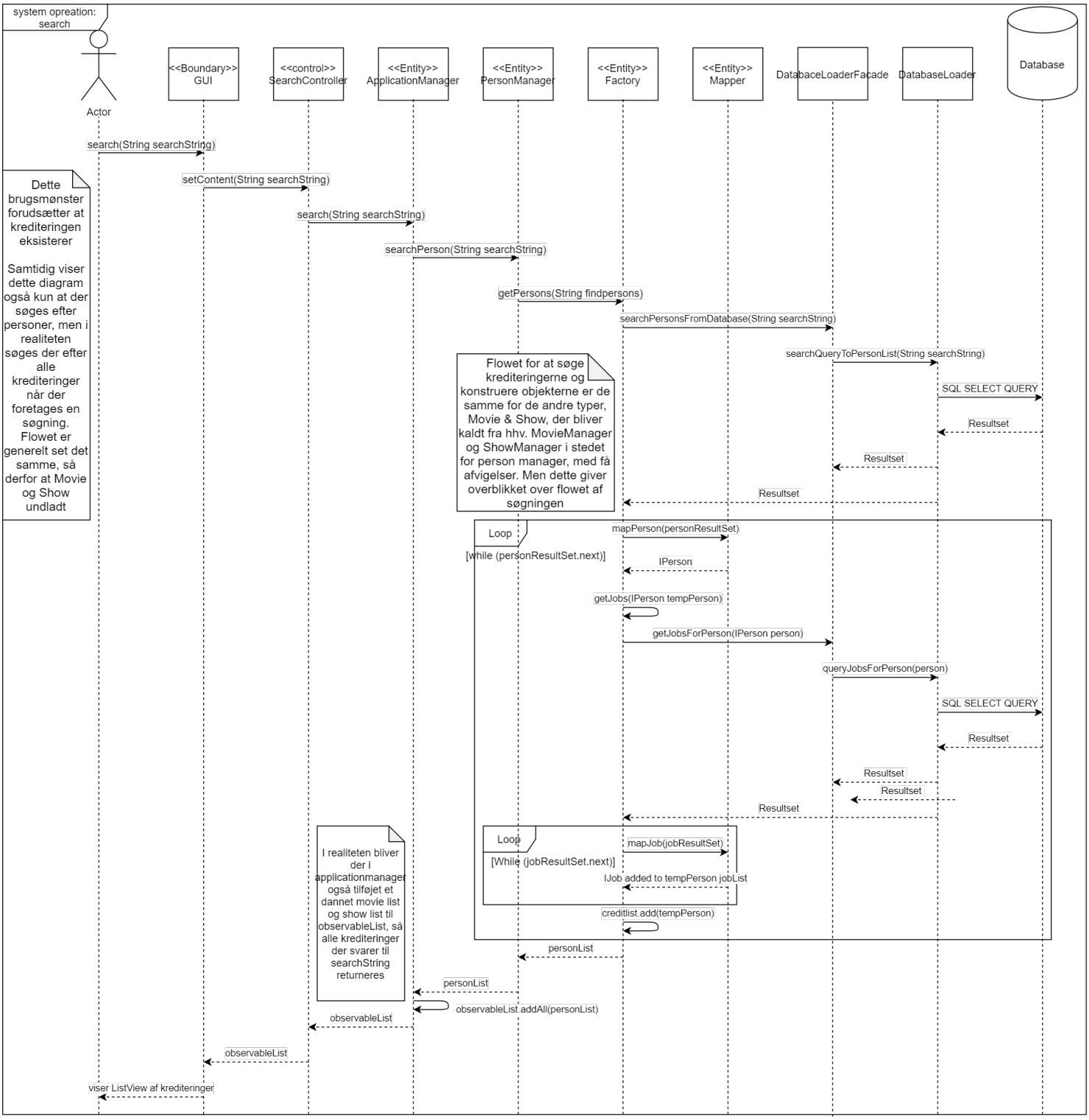
Kontrakt	
Operation	search(searchString)
Refererer til	Brugsmønster: Foretag en søgning
Ansvar	Ansvaret for denne operation er at at modtage en forespørgsel fra en bruger og vise de krediteringer der svarer til forespørgslen, for brugeren hvis: • Der eksisterer krediteringer der matcher brugerens forespørgsel • Den ønskede kreditering er godkendt af TV 2 moderatorer Den liste som brugeren får vist, vil indeholde en mængde af krediteringer der passer til deres forespørgsel, og kunden kan her vælge en af disse krediteringer, og få yderligere information om den
Prækonditioner	Datebasen indeholder data
Postkonditioner	Brugeren får vist en liste af krediteringer som matcher forespørgslen.

Tabel 11: Operationskontrakt for operationen search

Tabel 11 fortæller at ansvaret for search operationen er at finde de krediteringer der matcher brugerens forespørgsel, og returnere dem så de kan ses. Dette kræver at der er data det matcher i databasen som samtidig er godkendt af en TV 2 moderator. Herefter udarbejdede gruppen et sekvensdiagram for operationer inde i systemet. Først på figur 6 er resultatet for iteration 1, og derefter på figur 7 resultatet for iteration 2



Figur 6: Sekvensdiagram for operationer på ved searchAndGetCredit i iteration 1



Figur 7: Sekvensdiagram for operationer på ved search

Operations sekvensdiagrammet på figur 7 viser hvilke klasser som systemet går igennem for at lave en søgning. Det starter med at aktøren indtaster sine søgekriterier og trykker søg på GUI'en. Dette kalder funktionen search som tager en string som parameter. Search kalder funktionen setContent på SearchController klassen, som har til ansvar at få en liste fra searchString og vise den liste på GUI. SearchController kalder så funktionen search på application manager, som tager searchString og søger efter de forskellige typer af krediteringer. I dette eksempel vises kun søgningen efter personer, men flowet er det samme for film og serier, og derfor er de ikke medtaget i diagrammet. Application manager kalder searchPerson på PersonManager. PersonManager står for mange af funktonerne på objekter af typen Person. Personmanager kalder derefter getPersons på Factory. Factory er den klasse der står for at afgøre objekter der skal konstrueres og hvor data skal hentes fra. Factory kalder så searchPersonsFromDatabase på DatabaseLoaderFacade og sender igen searchString videre. DatabaseLoaderFacade er den simple indgang til databseLoader, hvor Facaden kalder funktionen searchQueryToPersonList stadig med searchString som parameter. Denne funktion queryer så databasen ved at finde de tupler der minder som søgekriteriet searchString. Databasen returnerer derefter et ResultSet af tupler, som sendes tilbage til Factory gennem DatabaseLoader, og DatabaseLoaderFacade. Her begynder konstruktionen af objekterne. Factory starter et loop, der slutter når der ikke er flere tupler i ResultSettet. For hver tupel, kalder Factory mapPerson på Mapper og sender en tupel som parameter. Metoden mapPersons tager denne tupel, og instantierer et person-objekt som en IPerson. En produktion som denne person har været med på kaldes et job. Factory sender den nyligt instantierede person til databasen (ved samme flow som perons) og får returneret et ResultSet af Jobs der har personens personID. Factory starter igen et loop over alle tupler i ResultSettet og tilføjer et instantieret job til en Arrayliste som sættes til personens jobs variabel. Her er personen konstrueret færdig, og personen tilføjes til et liste af krediteringer, creditList. og loopet kører igen. Når der ikke er flere tupler, returneres creditList, først gennem PersonManager, til ApplicationManger der tilføjer listen med personer til en generel list af Krediteringer. Her vil ApplicationManager så søge efter de nadre typer af krediteringer, film og serier, og tilføje dem til listen også. Når dette er gjort, laves listen om til en ObservableList som returneres til SearchController som sætter et ListView på GUI til at vise denne Observable list, og her får aktøren så vist listen af krediteringer der svarer til deres søgning.

4.4 Design

Der er fra kravspecifikationen stillet krav om at system skal bestå af en tre-lags arkitektur. Dette krav et også prioriteret som et "must"krav.

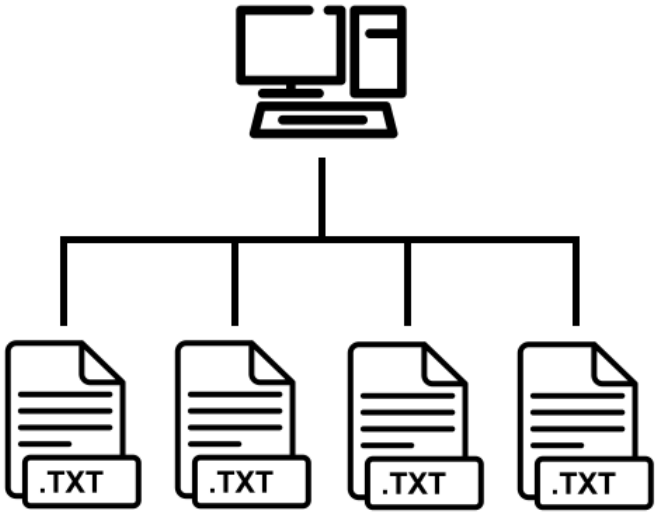
4.4.1 3-lags arkiteturen

4.4.2 Skalérbarhed

4.4.3 Klassediagram

4.5 Database Design

Databasen bestod i første iteration af individuelle TXT filer som blev brug som CSV filer. Dette lod os dele dataen op i de forskellige typer af objekter vi gerne ville have opbevaret, f.eks. Episode, Person og Movie, så hver type fik sin egen fil. På den måde undgik vi at indlæse alt vores data når vi skulle have fat i noget specifikt, og dermed spare ressourcer i form af tid og lagring.



I anden iteration er der gjort brug af en SQL relationel database, på baggrund af kravene til anden iteration.

4.6 Implementering

4.7 Test

5 Diskussion

6 Konklusion

7 Perspektivering

8 Procesevaluering

Referenceliste

[1] Ukendt-forfatter, "Kantar seer-undersøgelse," Last accessed 26 May 2021. http://tvm.tns-gallup.dk/tvm/pm/2020/pm2003_Consolidated.htm.

[2] T. og SDU, "Credits management, a case by tv 2 denmark a/s," Last accessed 26 May 2021. https://docs.google.com/document/d/1p6lQjWV76TX9uTLst2OAdmV07XfMRn5fObelzBpd2EI/edit.

A Oversigt over kildekode

B Brugervejledning

C Samarbejdsaftale

D Vejlederaftale

E Projektlog

F Udfyldt rapportkontrolskema

Kapitel	krav	Opfyldt +/-
Forside	Projekttitel, uddannelsesinstitution, fakultet, institut, uddannelse, semester, kursuskode, projektperiode, vejleder, projektgruppe og projektdeltagere (fornavn, efternavn, sdu-email). Må gerne have illustrationer.	
Titleblad	Samme oplysninger som på forsiden, samt afleveringsdato og projektdeltagernes underskrifter (Projektdeltagernes aktive deltagelse i projektforløbet anerkendes gensidigt ved projektdeltagernes underskrifter). Må ikke have illustrationer.	
Resumé	• En kort introduktion til projektet - hvad blev der arbejdet med og hvorfor. • Problemformuleringen og vigtige afgrænsninger. • Metode - hvordan angreb I problemet og hvordan realiserede I løsningen (hvem, hvad, hvornår og hvorfor) • Hovedresultater og konklusioner – hvad kom der ud af arbejdet. (max 1 side)	
Forord	Hensigten med rapporten, målgruppe, forhistorie, anerkendelser.	
Indholdsfortegnelse	Samlet indholdsfortegnelse for hele projektrapporten. Højst to eller tre niveauer i indholdsfortegnelse (der kan evt. være flere i selve rapporten). Afsnit på niveau 1 og 2 skal være nummererede.	
Læsevejledning	Vejledning i hvordan rapporten kan læses, eksempelvis i form af hvilken rækkefølge afsnittene kan læses i, og hvordan sammenhængen er mellem de forskellige dele af rapporten, herunder mellem hovedrapport og bilag. Rapportens målgruppe.	

<table>
<tr><td>Redaktionelt</td><td>Skriveprocessen og ansvarsområder i skriveprocessen.
Ansvarsområder kan fx beskrives på fx følgende form:
<table><tr><th>Afsnit</th><th>Ansvarlig</th><th>Bidrag fra</th><th>Kontrolleret af</th></tr><tr><td>Afsnit a</td><td>Person a</td><td>Person b</td><td>Person a, b, c</td></tr><tr><td>Afsnit b</td><td>Person b</td><td>Person a</td><td>Person a, b, c</td></tr><tr><td>Afsnit c</td><td>Person c</td><td>Person b</td><td>Person a, b, c</td></tr></table></td><td></td></tr>
<tr><td>Indledning</td><td>Projektets rammer og baggrunden for projektet. Resume af udleverede case. Problemformulering og afgrænsninger. Formål og mål med projektet. Problemformulering og afgrænsninger. (indledningen må gerne includere materiale direkte fra inceptionsdokumentet, det skal blot have en tydelig reference)</td><td></td></tr>
<tr><td>Faglig vidensgrundlag</td><td>Begrebsdefinitioner, teori og fagligt vidensgrundlag.</td><td></td></tr>
<tr><td>Metode og planlægning</td><td>Metode: Benyttede metoder i projekt (hele projektet). Kombination af UP og Scrum. Fordele og ulemper</td><td></td></tr>
<tr><td></td><td>Planlægning: Plan for elaborationsfasen og de enkelte iterationer. Prioritering af krav i planlægningen. Det faktiske udviklingsarbejde. Faserne, iterationerne og det faktiske arbejde i dem? Scrum: backlogs, roller, begivenheder, scrum-buts.</td><td></td></tr>
<tr><td>Hovedtekst
(skal indeholde resultater både fra iteration 1 og fra iteration 2)</td><td>Overordnede krav: Opdateret resume af overordnede krav fra inceptionsdokument inklusive overordnet brugsmønsterdiagram og oversigt over supplerende krav.</td><td></td></tr>
<tr><td></td><td>Detaljerede krav: Detaljeret brugsmønsterdiagram (hvis relevant). Detaljerede brugsmønsterbeskrivelser. Detaljerede beskrivelser af supplerende krav, fx organiseret efter FURPS+.</td><td></td></tr>
<tr><td></td><td>Analyse: Overvejelser, beslutninger og resultater vedr. analysemodellen, inklusive både den statiske og den dynamiske side af analysemodel.</td><td></td></tr>
</table>

	Design: Overvejelser, beslutninger og resultater vedr. Softwarearkitektur og detaljeret design, herunder design af persistens.	
	Databasedesign: Overvejelser, beslutninger og resultater vedr. tabeldesign og SQL-forespørgsler.	
	Implementering: Overvejelser, beslutninger og resultater vedr. konvertering fra design til kode illustreret gennem udvalgte centrale eksempler, samt andre vigtige implementeringsbeslutninger. Implementering af database.	
	Test: udførte test samt resultatet af dem.	
Diskussion	Hvad er der opnået og hvad er der ikke opnået i projektet i forhold til det forventede som beskrevet i indledningen. Hvad er styrkerne og svaghederne ved resultaterne. Kunne I have opnået bedre resultater?	
Konklusion	Opsummering af resultaterne og diskussionen af dem. Svar på problemformuleringen.	
Perspektivering	Er den fundne løsning brugbar i anden sammenhæng? Hvad bidrager løsningen og den opnåede viden til. Fremtidigt arbejde (næste skridt i projektet, hvis I havde mere tid).	
Procesevaluering	Processen og gruppens refleksion over processen: Læringsprocessen, teamroller, samarbejdet internt i gruppen og med vejleder, projektarbejdsformen, arbejdsformer, metoder, skriveprocessen, den tidsmæssige styring af projektet,ledelse af projektet, arbejdsfordeling i projektet m.m. Hvordan ville I gribe arbejdet an, hvis I skulle starte forfra?	
Referenceliste	Litteratur angivet på en anerkendt form. (Alle former for litteratur som bøger, artikler og hjemmesider) Kildehenvisninger i teksten. Materiale som gruppen ikke selv har fremstillet i dette projekt skal være angivet med kilde! Alle kildehenvisninger i teksten skal være anført på samme måde. Kildeangivelser på figurer, grafer etc. som projektgruppen ikke selv har frembragt.	

Bilag	A. Oversigt over kildekode B. Brugervejledning C. Samarbejdsaftale D. Vejlederaftale E. Projektlog F. Udfyldt rapportkontrolskema G. Inceptionsdokument I. (Andre Bilag)	

Rapporttekniske elementer		
Layout	Er der anvendt samme layout i alle kapitler. Er layout overskueligt/harmonisk.	
Sprog	Formidler rapporten projektet faglig og sagligt. Er sproget neutralt, aktivt, upersonligt, konkret, præcist, kortfattet og korrekt. (Procesevalueringen må benytte personligt sprog)	
Sidenummerering	Er der korrekt og konsistent sidenummerering i rapporten.	
Figurer/diagrammer	Er alle figurer konsekvent nummererede. Er der figurtitel og figurtekst til alle figurer. Er figurtitler og figurtekster dækkende og afklarende. Er figurerne tydelige og læsbare. Er figurerne informationsgivende og i den rette sammenhæng.	
Tabeller	Er alle tabeller konsekvent nummererede. Er der en forklarende tabeltekst til alle tabeller. Er alle søjler og rækker forsynet med parametre. Er der enheder på alle relevante rækker og søjler.	
Sporbarhed af begreber	Er der en konsekvent brug af samme betegnelse for et givet begreb igennem rapporten.	

G Inceptionsdokument

SDU
University of
Southern Denmark

Projekt 2 - TV 2

2. Semester

Titel:	Udvikling af Softwaresystemer
Institution:	Syddansk Universitet, Det tekniske Fakultet
	Mærsk Mc-Kinney Møller Instituttet
	Campusvej 55, 5230 Odense M
Uddannelse:	Software Engineering
Semester:	2. Semester, F21
Kursuskode:	T510043101
Omfang:	10 ECTS
Vejleder:	Henrik Lykkegaard Larsen
Projektgruppe:	SE-04
Projektperiode:	Februar 2021 til Juni 2021

casti19, haped20, vbruu20, jedie20, jobel20, perat17
Casper Stillinge, Hans Pedersen, Victor Bruun
Jesper Diederichsen, Jonas Beltoft, Peter Ratgen

May 27, 2021

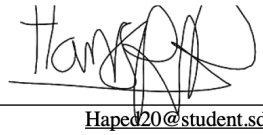
Titelblad

Titel:	Udvikling af Softwaresystemer
Institution:	Syddansk Universitet, Det tekniske Fakultet
	Mærsk Mc-Kinney Møller Instituttet
	Campusvej 55, 5230 Odense M
Uddannelse:	Software Engineering
Semester:	2. Semester, F21
Kursuskode:	T510043101
Projektperiode:	F21
Omfang:	10 ECTS
Vejleder:	Henrik Lykkegaard Larsen
Projektgruppe:	B-04

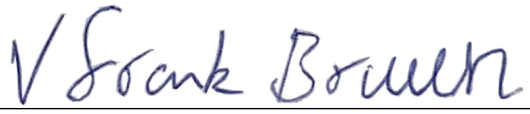
Bidragserklæring

Ved at underskrive dette dokument bekræfter hvert enkelt gruppemedlem, at alle hæfter kollektivt for dokumentets indhold, samt at alle har bidraget til projektets udførelse.

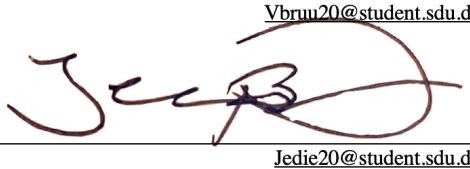
Casper Stillinge — Casti19@student.sdu.dk



Hans Pedersen — Haped20@student.sdu.dk



Victor Bruun — Vbruu20@student.sdu.dk



Jesper Diederichsen — Jedie20@student.sdu.dk

Jonas Beltoft — Jobel20@student.sdu.dk

Peter Ratgen — Perat17@student.sdu.dk

Indholdsfortegnelse

1 Indledning

Når et TV-program er slut, skal der altid være rulletekster, disse er begrænset til 30 sekunder, der er derfor ikke altid plads til at vise alle personer der har været med i en produktion. Ved at undgå at vise disse 30 sekunders rulletekster med krediteringer, kan man bruge pladsen til at vise reklamer. Ved ikke at vise rulletekster kan TV 2 øge dets omsætning med op til 60 millioner DKK. Derfor har TV 2 brug for et system til at styre krediteringer på forskellige produktioner. Dette inkluderer at man skal kunne tilføje nye krediteringer i systemet.

Udover at rulleteksterne tager de 30 sekunder at vise, kræver de nuværende systemer at medarbejdere på TV 2 skal bruge tid på et skrive krediteringer ind manuelt i deres eksisterende systemer, så som EPG-systemet (Electronic Programming Guide). På nuværende tidspunkt modtager TV 2 krediteringsoplysninger via tekstdokumenter, fremsendt fra producenterne. Med et system til kreditering, vil TV 2 kunne fokusere på at kvalitetsikre data i systemet. Her vil producenterne stå for at indskrive krediteringer i systemet.

TV 2 og SDU har givet en række krav som en del af den givne case-beskrivelse. Under disse krav er der nogle som skal være implementeret i det endelige produkt. Derudover disse så er der nogle af kravene som har mindre betydning i det endelige produkt, og derfor kan udelades i starten. Krav der skal være med i det endelige produkt er muligheden for at tilføje krediteringer, samt inklusionen af et brugersystem.

Formål og mål Når inceptionsdokumentet er komplet, har gruppen et veldefineret projektgrundlag med krav, scope og problemformulering samt hvilke metoder vi vil anvende for at løse den fundne problemformulering. Et andet formål med inceptionsdokumentet er at afdække om der er grundlag for en business case.

Problemanalyse Målet med dette afsnit er at analysere den stillede case fra TV 2 og identificere mulige problemer som skal adresseres i udførelsen af dette projekt. Da TV 2 har stillet en relativt udpenslet case, i denne har TV 2 beskrevet flere problematikker omkring krediteringen af personer relateret til produktioner. Denne information struktureres i et problemtræ denne ses på figur 1.

Der findes regler omkring at personer skal krediteres. Dette gælder alt, hvad der vedrører sig et program, hvadenten det er grafik eller musik. TV 2 specificerer at personer krediteres grundet forskellige årsager. Disse er:

- Opretshavsloven
- TV 2s overenskomster
- Kontrakter for ophavsmænd og udøvende kunstnere
- Almindelig god skik

Der skal krediteres, selvom rettighederne til materialet er erhvervet. Både personer og firmaer skal krediteres, firmaer krediteres i slutningen af rulleteksterne. Rulletekster med krediteringer er begrænset til 30 sekunder.

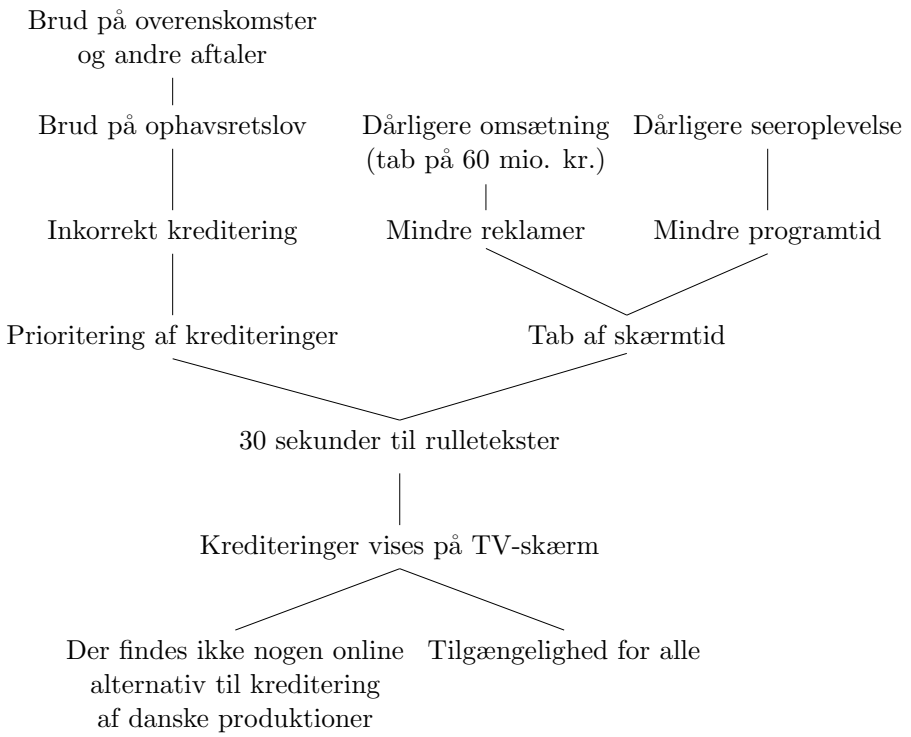


Figure 1: Problemtræ omkring de informationer, der er givet i casen.

Når problemtræet betragtes ses det, at de 30 sekunder, som det tager at vise rulletekster, har nogle afledte effekter både i form at prioritering af krediteringer, samt tab af skærmtid. Årsagen til dette problem er at krediteringerne vises på TV. Hvis dette ikke var tilfældet, ville virkningerne ikke være de samme. Vi betragter det at krediteringerne vises på TV som vores hovedproblem.

Problemformulering Hvordan kan vi udvikle en prototype til et krediteringssystem, der vil kunne erstatte de klassiske rulletekster efter et afsluttet program?

Dette leder til nogle underspørgsmål:

- Hvordan er reglerne for krediteringer for danskproducerede programmer?
- Hvilke informationer skal og må inkluderes i krediteringer?
- Hvordan kan sådan et program gøres mest mulig brugervenligt(tilgængeligt) for både seer og administrator?
- Hvordan kan vi sikre et velstruktureret program, der mindsker fejl og ventetid samt strukturere databasen således, at der undgås fejl, duplikering af værdier og null-værdier?

Afgrænsning Tjenesten til håndtering af TV 2s krediteringer bliver afgrænset til danske programmer og shows, dvs. at udlandske udsendelser ikke er omfattet af dette system og derfor stadig vil køre med alm. rulletekster. Produktet der udarbejdes produceres som en prototype. Gruppens produkt ville altså ikke blive lavet til et fuldendt system, der løser alle de givne krav. Krediteringen skal også overholde alle de givne regler for kreditering.

2 Fagligt vidensgrundlag

2.1 Kravudvikling

Når kravene til et system skal findes vil man som regel følge et workflow, der sikrer at kravene bliver så præcise, komplette og konsistente som muligt.

- Præcise krav, da tvetydige krav kan fortolkes på forskellige måder.
- Komplette krav, så de indeholder beskrivelser af alle aspekter.
- Konsistente krav der er konsekvente. Kravene må altså ikke modsige hinanden eller være i konflikt med hinanden.

Kravene til et softwaresystem er derfor vigtige at have styr på, da de fastsætter hvad systemet skal gøre, og definerer begrænsninger for dets udvikling og drift.

For at få et optimalt workflow til at få skabt nogle gode krav, har vi i gruppen kigget på at lave en use case-model.
Det indebærer, at vi først har fundet de aktører, der forbindes med systemet. For at identificere en aktør kan man spørge: "Hvem eller hvad bruger systemet?" "Til hvad?" og "Hvorfor?". Man kan kigge på hvilken rolle de spiller i interaktionen med systemet. Man kan også spørge, hvad andre systemer bruger eller bidrager med til systemet. Når man er afklaret med hvilke aktører der er, skriver man dem op i listeformat, hvor man inkluderer aktørernes navne, deres mål (hvad og hvorfor) og deres bidrag.

Næste punkt i en use case-model er at finde og identificere use cases. Når man identificerer use cases vil man gå ud fra listen over aktører, og stille sig selv en række spørgsmål. Hvilke mål har en aktør? Det er skrevet ned i listen, og man kan derfor lave en use case til hvert mål aktøren har. Hvilke funktioner vil en specifik aktør have af systemet? Til det kan man lave en use case til en funktion, der giver målbare resultater til aktøren. Gemmer systemet information, som den henter frem? I så fald for hvilken aktør gør den det? Dette er også værd at overveje, når vi skal lave use cases. Ligeledes som med aktørerne, laver vi en liste over use cases, der indeholder deres navne og deres formål.

Til slut skal der identificeres systemgrænser. Til det skal der først tegnes et use case diagram med aktører, de fundne use cases, og linjer herimellem, der forbinder aktøren med de relevante use cases. Nu er det vigtigt, at man evaluerer, sine use cases, aktører og emnet. Ud fra sine evalueringer kan man nu danne sig et billede af, hvilke krav, der stilles til systemet, og man ender derfor med en skitsering af en kravspecifikation til systemet.

2.2 Objekt-orienteret programmering

Hvad er OOP? Objekt-orienteret programmering (eller OOP), er et programmeringsparadigme baseret på et koncept af "objekter". Disse objekter kan indeholde en række data, generelt kaldet attributter. Kode kan være skrevet i form af en procedure, der skal lave noget bestemt, dette bliver kaldt en metode. Et eksempel kunne være at kode en person, denne person ville være et objekt. Personen har flere forskellige metoder til at kunne tale, bevæge sig, trække vejret OSV. Person objektet skal også have nogle attributter så som en højde, deres køn, deres alder osv.

Ved brug af Java-kodesprog i dette projekt kommer vi til at gøre stor brug af klasser. Når man definerer en klasse, opretter man egentlig et blueprint til et objekt. Heri defineres

ikke direkte data, men der defineres hvad der skal gøres med noget data der er givet til objektet.

2.2.1 OOP's fire basale koncepter

Indkapsling Når man snakker om indkapsling inden for OOP, så betyder det at nogle data eller noget funktionalitet bliver indkapslet. Dette gøres ved hjælp af Access modifiers. Access modifiers hjælper til at bestemme, hvem der har adgang til data og funktioner, og hvor meget der er adgang til. Der findes en række forskellige access modifiers som kan ses på tabellen 1 på side 4. Indkapsling er en god måde at lave dataskjulning for andre klasser. Indkapsling gør det også nemmere at teste produktets sikkerhed.

Access Modifier	Inden for klassen	Inden for pakken	uden for pakken kun af subklasser	uden for pakken
Private	X			
Default	X	X		
Protected	X	X	X	
Public	X	X	X	X

Table 1: Liste over tilgange ved de forskellige access modifiers

Abstraktion Abstraktion er en måde på at gemme specifikke værdier og metoder. Abstraktion kan ses som en udvidelse på indkapsling, siden det kan gemme information eller metoder for ekstern kode. Dette gør interfacet af objekter meget simpelt. F.eks. kan abstraktion forstås ved at kigge på kroppen, vores skind fungere som an abstraktion til at gemme hvad der foregår inde i kroppen.

Nedarvning Nedarvning kommer fra at være en subklasse. Her nedarver subklassen attributter og metoder fra dens superklasse. Dette gøre det muligt at nemmere kunne genbruge kodestykker. Derudover bruges det også til metodeoverskrivning så man kan opnå runtime polymorfi.
Imodsætning til nedarvning, hvilket er it såkaldt "IS-A" forhold, findes der også et "HAS-A" forhold, et aggregat. Et HAS-A forhold beskriver de gange hvor et objekt gøre brug af et andet objekt til at gemme data. F.eks. hvis man har en kunde med nogle informationer inde i et objekt, så kan der herunder oprettes et adresse objekt hvilket indeholder information som by, kommune, post nummer osv.
Et aggregat kan være god at bruge i situationer hvor en stor klasses information kan deles ud i mindre bidder. Det hjælper også meget på kode genanvendelse. I den virkelige verden kan vi forestille os nedarvning ved at kigge på insekter. Alle insekter har lignende egenskaber så som at have seks ben eller et exoskelet. Altså ville en grasshoppe eller myre have nedarvet disse egenskaber fra superklassen af insekter.

Polymorfi Polymorfi er muligheden for at et objekt kan tage flere former. Dette gør at vi kan implementere en metode fra en superklasse i en subklasse på forskellige måder.
Det ville altså sige at hvilket som helst subklasse objekt i Java kan hvilken som helst form som er i superklassens hierarki, inkluderende sig selv. Generelt er der to typer af polymorfi:

- dynamisk polymorfi -
  også kendt som run-time polymorfi. Dette kan beskrives ved at tænke på superklasse-subklasse forhold. De begge har den samme metode, og subklassen har overskrevet

superklassens metode (metode overskrivelse).
Når et objekt bliver tildelt en klasse reference, og en metode fra objektet bliver kaldt, så er det metoden indeni objektets klasse, der bliver udført. Der bliver altså ikke kaldt metoden fra reference klassen, hvis altså referencen er en superklasse.

- statisk polymorfi -
  også kendt som compile-time polymorfi eller metode overbelastning. Dette betyder, at man har angivet flere af de samme metodenavne, men med forskellige argumenter. Det betyder altså, når man kalder sådan en metode, så ville kompilatoren beslutte hvilken metode, der bliver kørt i forhold til hvilke parametre, der er angivet.

2.3 Trelags arkitektur

Trelags arkitekturen er en software arkitektur, der deler programmet op i tre lag. Et præsentationslag, et logiklag og et datalag:

- Et præsentationslag som udelukkende står for at forsyne brugeren med en brugerflade og hente information fra brugeren og med den information kommunikere med det næste lag, logiklaget.
- Logiklaget står for alle de logiske operationer som er associeret med de gældende elementer i præsentationslaget på baggrund af den information, som brugeren har indtastet. Logiklaget kommunikerer også med datalaget, og kan her sammenholde information fra brugeren med det i databasen for at tage en beslutning om, hvad programmet skal gøre.
- Datalaget står for at lagre systemets data som logiklaget kan anvende.

For en succesfuld anvendelse af trelags arkitekturen vil brugeren udelukkende blive præsenteret for præsentationslaget og behøver ingen viden om de underliggende lag, da al kommunikation mellem bruger og program sker i dette lag. Brugen af trelags arkitekturen giver nogle fordele.

- Først og fremmest kan udarbejdelsen af de forskellige lag uddelegeres således at de udvikles samtidig. Dette gør udviklingen af programmet hurtigere
- Da al logikken sker et sted, bliver programmet mere skalérbart idet at koden/programmet ikke er kludret sammen og har for mange indgangspunkter.
- Hvis et lag bryder sammen eller ikke virker, påvirkes de andre ikke, da de er opdelt.
- Grundet at trelags arkitekturen opdeler præsentationslaget og datalaget med et logiklag, vil man med korrekt implementering kunne sikre, at en bruger ikke får tilladelse til at tilgå data, som ikke er tiltænkt dem.

2.4 Unified Process

Unified Process (UP) er en iterativ og trinvis udviklingsproces. UP er delt op i 4 trin, eller faser, disse og deres definitioner er:

- Inceptionsfasen

  I inceptionsfasen skal man finde ud af projektets gennemførlighed, finde frem til vigtige krav og identificere potentielle risici.

- Elaborationsfasen

  I elaborationsfasen skal der laves en arkitekturprototype, oveni at risikovurderingen forbedres. Hertil skal hovedparten af brugsmønstrene findes, samt at der skal lægges en plan for konstruktionsfasens forløb.

- Konstruktionsfasen

  I konstruktionsfasen færdiggøres den endelige identifikation af brugsmønstre, samt disses beskrivelse og realisering. Analyse , design, implementation og test færdiggøres også i denne fase. Undervejs redigeres projektets risikovurdering.

- Transitionsfasen

  I denne fase rettes fejl, og brugerne forberedes på at anvende softwaren. Her laves også manualerne og anden dokumentation, til sidst gennemføres et review af projektet.

En vigtig pointe omkring UP er at det er en proces drevet af brugsmønstre (use cases), samt at der fokuseres på risikostyring og arkitektur. Og som tidligere nævnt er der fokus på iterationer og en trinvis udviklingsprocess. Dette betyder at systemet udvikler sig trinvist for hver iteration.

2.5 Scrum

Scrum sprint er en agil metode der fokusere på forberedelse, eksekvering og idriftsættelse, af små dele af et projekt. I Scrum tales der om 3 hoved roller, den første er Scrum Team som i vores tilfælde kan være en gruppe udviklere. Derefter har vi en Product Owner, som vil være en repræsentant for det firma der har bestilt projektet. Til sidste har vi en Scrum Master som overordnet set er den der står for at planlægge og at de forskellige aktiviteter i Scrum bliver overholdt. Dernæste indeholder Scrum metoden en række artefakter, hvor de tre vigtigste er følgende.

- Product Backlog er en liste over de prioriteret krav.
- Sprint Backlog er en delmængde af de opnåede krav som står færdigt efter et sprint.
- Burndown Chart er en visualisering af hvor meget arbejde teamet mangler før projektet er færdigt. [1]

2.6 UML

UML eller Unified Modeling Language, tilbyder en måde at visualisere et systems arkitekturelle blueprints i et diagram.

2.6.1 Klassediagrammer

Det er bl.a. brugt til at lave Class Diagrams, da det giver en oversigt over hvilke klasser der er i et program, hvilke attributter og metoder de har, samt disses synlighed (Access Modifiers). UML anfører også sammenhængen mellem klasserne, om de fx nedarver fra klassen eller andet, hvilket er en stor del af grunden til at man bruger UML. Ud fra det kan man aflæse hvordan forskellige klasser bliver brugt andre steder i programmet, og derved få en bedre forståelse for hvordan programmet virker, uden at man har set programmet køre. Der er særlig notation til at fremhæve alle OOP's fire basale koncepter,

hvilket gør det muligt at udarbejde og/eller visualisere præcist det program, der er ønsket. Øverst ses navnet på den klasse, der beskrives. I sektionen under, er alle klassens attributter, og under dem er klassens metoder. Hvis klassen har en eller flere constructors, kan disse også ses, da de har det samme navn som klassen. På grund af alt dette, kan UML være et meget kraftfuldt værktøj under udviklingen af et objektorienteret program, da man kan visualisere alle de objekter og klasser, der bliver lavet og instantieret. Samt sammenhængen mellem disse objekter og klasser. UML bruges også til andre ting, såsom brugsmønster-diagrammer, som giver et overblik over aktører og de brugsmønstrer de interagerer med. Dette er en anderledes måde at modellere på, men det er alt sammen under UML, da det er en universel måde at forbinde elementer på, i en visuel kontekst.

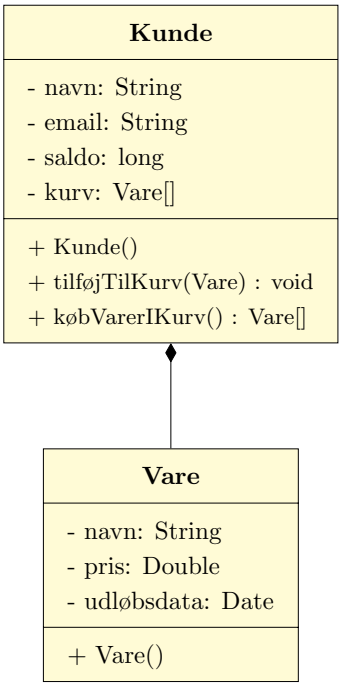


Figure 2: UML klassediagram for sammenhængen mellem en Kunde og en Vare

2.6.2 Beskrivelse af database med UML

UML kan også bruges til at vise en database. Det ses nemt hvilken udgave, der er brugt, da man her vil se notationer så som {PK} som beskriver at denne attribut er en primær nøgle i denne tabel. Hver kasse er her ikke en klasse, men en tabel. Den har kun attributter, da en tabel ikke har funktioner. Et UML diagram over en database vil også have forskellige måder at visualisere sammenhæng mellem tabellerne. Klassediagrammet som beskrevet har kun én standardiseret måde at forbinde klasserne på, hvilket gør det nemmere at lære og hurtigere at finde ud af, hvad der menes med de forskellige symboler man støder på, hvorimod et UML diagram over en database har flere forskellige måder at vise sammenhæng på, selvom de viser det samme. I klassediagrammet vil man se symboler som f.eks. betyder at denne klasse nedarver fra en anden klasse, eller bl.a. viser at den ene klasse "bruger" eller "afhænger" af en anden klasse. Dette finder man ikke i et database UML diagram. Her viser forbindelserne, hvor mange elementer, der kan være, f.eks. kan en kunde købe mange varer, men en type vare kan købes af mange forskellige kunder, så det vil kaldes en mange til mange forbindelse. Dette kan noteres på forskellige måder, f.eks. med crows foot, Chen eller andre. Vi som gruppe vil bruge Chen, som er en meget simpel måde at vise, om det er 0, 1 eller flere. Disse notationsformer er også brugt i ER og EER modeller, men det betyder ikke at et UML diagram er det samme som et ER eller EER diagram, da de ikke går direkte ned i databasen, men er mere en overordnet beskrivelse og kan vise andre ting end UML kan.

2.7 MoSCoW

MoSCoW er en model der bruges når en liste af krav skal prioriteres. Kravene indeles i 4 grupper, efter hvor vigtige de er for projektets succes. Her startes der med "Must Have" som er de vigtigste krav og som er nødvendige for at projekt bliver en succes. Dernæst kommer "Should Have" som er krav, der ikke er nødvendige, men det vurderes at de giver en tilstrækkelig værdi til projektet, i forhold til arbejdsindsatsen. "Could Have" er krav som ikke er vigtige for projekts gennemførsel, de kan tages med i tilfælde af overskydende arbejdsressourcer, men der planlægges ikke efter, at de skal medtages. Til sidst findes

"Would Have" som er krav, der giver mening for projektet, men er valgt fra i den pågældende udgave af projektet.

2.8 FURPS+

- Usability

  "Usability" handler om de menneskelige interaktioner med programmet. Det gælder om at se på, hvor effektivt programmet er. Er dokumentationen for programmet i orden og uddybende nok, til at forklare hvad programmet står for.

- Reliability

  "Reliability" omhandler det, der vedrører oppetid, præcision i systemets beregninger.

- Performance

  "Performance" drejer sig om, hvordan systemet skal reagere på større mængder af information.

- Supportability

  "Supportability" omhandler krav, som vedrører testbarhed, kompatibilitet, konfigurerbarhed.

- Plus

  "Plus" er den del af FURPS+ der indeholder begrænsninger og specifikke krav for systemet.

  - Design constraints

    Man kigger på designmæssige begrænsninger, der kunne være for projektet. Ændrer I/O enheder eller den valgte DBMS, hvordan softwaren skal bygges?

  - Implementation requirements

    Man kigger også på implementationen og dets krav, og beskriver, hvordan programmørene skal forholde sig. Skal de bare forholde sig til standarderne?

  - Interface requirements

    Interface krav kigger på om, der er nogle interfaces som programmet skal fungere på eller med.

  - Hardware requirements

    Til sidst er der hardware krav, hvor der kigges på, om der er nogle krav til hvilken hardware systemet skal interagere med eller kører på, og om det skaber nogle forudsætninger for produktionen.

2.9 Eksisterende løsninger

2.9.1 The Movie Database (TMDb)

TMDb er en brugerflade, som gør det nemt for enhver person at finde hvilken som helst film, serie, person. På hver film eller serie kan man gå ind og se alle de medvirkende, hvilket er nemt listet med deres rolle ved produktionen. Går man ind og finder en specifik person kan man nemt se personlig information eller man kan se, hvad personen har været med i. Inde på hver person kan man også lave an specificering af deres roller, altså om man kun ville se film eller serier. TMDb er bygget og skrevet af fællesskabet der bruger systemet. Altså alt information der findes på siden er tilføjet af brugere fra hele verden.

2.9.2 Internet Movie Database (IMDb)

IMDb er en online database af information på en hel række platforme. Udover hvad nævnet fortæller om, at der kan findes af information om film, så har de også information om hjemme-videoer, TV-programmer, video-spil og online streaming, samt meget mere.
IMDb indeholder over 400 milioner data objekter, hvoraf op til 7.5 milioner titler, over 10.4 milioner individuelle personaliteter og over 11 milioner billeder. IMDb har altså en bred kundekreds. Deres database system er kendt af mange og er altid en af de første links på diverse søgemaskiner, når der søges efter noget som helst, de har i deres database. Deres hjemmesider byder på rigtig meget information meget hurtigt og nemt, men det kan også være for meget information.

3 Overordnet kravspecifikation

Til udarbejdelsen af projektets krav benyttes en brugsmønster-fokuseret fremgangsmåde med hensigt på at lave et brugsmønster-diagram. Brugsmønster-diagrammet skal medvirke til at visualisere, hvordan systemet vil virke i praksis, ved at vise relationerne mellem brugsmøsnterne og aktørerne.

3.1 Aktørliste

Før vi kan lave et uddybende brugsmønsterdiagram skal vi først og fremmest identificere aktørerne.

Aktører	**Beskrivelse**
Bruger	En almindelig bruger skal kunne søge og se krediteringer i systemet. Det vil sige de skal kunne: • Indtaste søgekriterier i et søgefelt – Kan søge efter personer – Kan søge efter programmer – Kan søge via et program id • Trykke søg • Få vist krediteringer for de indtastede søgekriterier • Ændre sprog • Kunne logge ind • Hente data ned som XML eller CSV fil

TV 2 Moderator	En moderator står for at kvalitetssikre de data som er blevet sendt ind af producenter: • Logge ind som Moderator • Fjerne krediteringer • Redigere eksisterende krediteringer • Godkende/afvise ny kreditering/ændring
Producent	En producent er en ansat hos det firma der laver et specifikt program og står for at indtaste krediteringer. Data bliver så tjekket igennem af en TV 2 Moderator: • Logge ind som producent • Oprette et show • Oprette en person • Redigere eksisterende krediteringer
Krediteret person	En krediteret person skal have mulighed for selv at indtaste mere personfølsomme data som en producent ikke har ret til grundet GDPR-mæssige årsager. Derfor kan vedkommende have mulighed for selv at tilføje det, hvis de ønsker
Systemadministrator	En systemadministrator skal kunne: • Logge ind som Systemadministrator • Fjerne krediteringer • Gennemse nylige ændringer • Tilføje og fjerne brugere som moderatorer • Fjerne et indlæg
Finansiel afdeling	Den finansielle afdeling er en sekundær aktør, der kan hente rapporter ud fra systemet, til brug af betaling af ophavsret
Samrådet for ophavsret / Producentforeningen	Denne sekundære aktør står for at acceptere ændringer, da de er foreninger, der står for at opkræve ophavsret
Loginsystem	Loginsystemet er en sekundær aktør, der står for at håndtere alle login. Det skal kunne differentiere mellem, om en bruger eller producent osv. logger ind. Det skal kunne integreres med TV 2's nuværende login til TV 2 Play, så en bruger kan benytte samme konto til begge tjenester

Table 2: Liste over aktører og handlinger de kan tage

3.2 Brugsmønsterliste

Med aktørene identificeret og deres mulige handlinger kortlagt bliver en liste over brugsmønstre formuleret.

ID	Brugsmønster	Beskrivelse
B01	Foretage en søgning	Brugeren går ind på hjemmesiden og bliver mødt af et søgefelt, hvor vedkommende skriver sine søgekriterier og bliver så mødt af nogle forslag, der minder om, det de har skrevet. Her kigger de dem igennem og bruger musen til at klikke på den person eller det program de vil ind på, og kan her se alt information, der er om et program eller en person.
B02	Oprettelse af person	En producent logger ind på hjemmesiden og vælger at oprette en ny person. Denne person tildeles alle de informationer der er påkrævet. producenten guides til udfyldningen via en formular/skabelon og skal blot udfylde de felter.
B03	Oprettelse af show	producent logger ind på hjemmesiden og udfylder en skabelon til oprettelse af show.
B04	Redigere krediteringer	Moderator logger ind på hjemmesiden og finder frem til den rette kreditering. Her trykker han rediger, og ændrer nu informationerne efter samme formular som skulle man tilføje en ny.
B05	Hente data ned som en fil	Har man foretaget en søgning og fundet det ønskede resultat, kan man vælge at downloade informationerne om programmet/personen/firmaet som en XML eller CSV fil.
B06	Valg af sprog	Brugeren skal have mulighed for at skifte mellem dansk og engelsk.
B07	Fjerne et indlæg	En TV 2 moderator og systemadministrator skal have muligheden for permanent at slette en person/firma/show. Det gøres ved at logge ind som moderator eller administrator og derefter finde frem til det indlæg, man har i tankerne. Herefter vælges sket og administratoren bedes bekræfte valget
B08	Oprettelse af moderator	Systemadminstratatoren skal have adgang til at oprette en moderator.
B09	Fjernelse af moderator	Systemadminstratatoren skal have adgang til at fjerne en moderator.
B10	Login	Inde på siden vælges login i menuen og her indtaster du informationer. Programmet checker så om informationerne matcher noget login i databasen, og hvis det gør, checker den derefter om du bliver logget ind som moderator eller systemadministrator
B11	Gennemse nylige ændringer i log	Systemadministratatoren skal have adgang til en log, hvor nylige ændringer i krediteringssystemet kan gennemses. Ændringer kan være tilføjelser, redigeringer eller fjernelser. Loggen skal fortæller noget om hvem, der har ændret, hvad der er blevet ændret og hvornår der er blevet ændret.

B12	Hente rapporter	Den finansielle afdeling skal have adgang til at hente rapporter på det data de skal bruge
B13	Acceptere ændringer	Samrådet for ophavsret / Producentforeningen skal kunne acceptere nylige ændringer, da det er de foreninger der står for registrering af ophavsret.
B14	Godkende/afvise ny kreditering/ændring	TV 2 Moderatoren skal have mulighed for at se nylige ændringer igennem og vurdere om de skal godkendes og tages med i systemet, eller afvises og redigeres

3.3 Brugsmønsterdiagram

Ud fra den brugsmønsterlisten, som gruppen har beskrevet ovenfor, laver vi et brugsmønsterdiagram. Se figur 3.3.

Figure 3: Brugsmønsterdiagram

3.4 Brugsmønster detaljeret

I dette afsnit vil vi gå i dybden med to brugsmønstre og udarbejde to detaljerede brugsmønstre på hhv. B01 foretage en søgning og B03 oprettelse af show.

Brugsmønster: Foretage en søgning
ID: B01
Primære aktører: Bruger
Sekundære aktører:
Kort beskrivelse: Brugeren bliver mødt af et søgefelt, hvor de indtaster deres ønskede søgekriterier og vælger herefter hvilken kreditering, de ønsker at få vist
Præconditioner: Brugeren har et søgekriterie i form af navn på show/person eller ID
Hovedhændelsesforløb: 1. Brugeren bliver mødt af et tomt søgefelt   • Brugeren kan vælge at logge ind 2. Brugeren indtaster et søgekriterie i form af navn på show/person eller ID 3. Brugeren søger med de givne søgekriterier 4. Brugeren får vist en liste af krediteringer, der matcher søgekriterierne 5. Brugeren vælger her den ønskede kreditering 6. Brugeren får nu vist den specifikke kreditering som de har valgt
Postkonditioner: Logiklaget har leveret informationer til brugeren, og brugeren har adgang til de krediteringsinformationer de ønsker
Alternative hændelsesforløb: • Skridt 5: Hvis brugeren ikke kan finde den ønskede kreditering kan de fortage endnu en søgning eller opgive

Table 4: Detaljeret brugsmønster over B01

Brugsmønster: Oprettelse af show
ID: B03
Primære aktører Producenter og firmaer (Opretter)
Sekundære aktører
Kort beskrivelse Producent for showet eller en SoMe-ansvarlig fra et firma kan logge ind på systemet og oprette showet, ved at udfylde en skabelon.
Præconditioner Man skal være logget ind som enten producent eller firma

Hovedhændelsesforløb
1. Producent eller firma logger ind på systemet. 2. Opretteren navigerer til siden med oprettelse af show. 3. Opretteren udfylder skabelon til oprettelse af show som indbærer: • Navn på show og evt. serie ID. • Dato for første udsendelse. • Navn på producent. • kategori. • mm. 4. Krediteret personer tilføjes til show. • Opretteren skal kunne hente personer, som eksisterer i databasen og tilføje dem til showet. • Ved tilføjelse af ikke eksisterende person: se alternativt hændelsesforløb. 5. Den udfyldte skabelon sendes til TV 2, som kvalitetstjekker, godkender og tilføjer til databasen.
Postkonditioner Showet skal godkendes af TV 2
Alternative hændelsesforløb Skridt 4. Ved tilføjelse af personer, der ikke allerede eksisterer i databasen, får opretteren mulighed for tilføjelse af den nye person ved et pop-up vindue, for yderligere specificering se B02 - Oprettelse af person.

Table 5: Detaljeret brugsmønster over B03

3.5 Supplerende krav

Udover brugsmønstrene har systemet også nogle supplerende krav, som bl.a. er stillet af TV 2

Oppetid	TV 2 har et krav om at systemet har en oppetid på 99,7 - 99,8% .
Svartid	TV 2 Ønsker at svartiden på systemet skal være så lav som muligt.
Sikkerhed	TV 2 stiller et meget stærkt krav om, at der er styr på, at al data i systemet er korrekt, da data skal anvendes for at sikre, at krediterede personer får den betaling, de har ret til.
3-lags arkitektur	Det ønskes at programmet følger en 3-lags arkitektur.
Unikke personer	Der er et krav om, at man skal kunne differentiere personer med f.eks. identiske navne.
Integration med andre systemer	TV 2 har et ønske om at programmet skal kunne integreres med en række af de systemer, de anvender i forvejen. Eksempler kunne være EPG og login-systemet til TV 2 Play.

Table 6: Supplerende krav

4 Kritiske risici

4.1 Krav

Definition af krav Det er en risiko at kravene ikke defineres korrekt, således at vigtige features går tabt. Det kunne også være en risiko at dårligt formulerede krav leder til at en feature eller funktion bliver implementeret forkert. En risiko er også at kravene ikke bliver udspecificeret nok til, at disse er krav er målbare.

Forebygning af risici Vi vil undgå dette ved at alle i gruppen har læst og godkendt kravene, samt nogle af kravene kommer direkte fra TV 2. Udover det er hele dokumentet blevet reviewet af 3 andre studiemedlemmer, som individuelt har læst og forstået disse krav. Uafhængigt af reviews, så har vi som gruppe overvejet ordvalgene i kravene meget omhyggeligt, så enhver risiko for misforståelse burde være minimeret.

4.2 Arkitektur

Valg af arkitektur Det er en risiko for projektet, hvis vi ikke finder den rette arkitektur at bygge systemet ud fra, således at vi ikke har noget godt udgangspunkt for projektet.

Det er vigtigt for TV 2 at have en høj oppetid, så hvis der konstrueres et system med en forkert arkitektur, er det svært at opretholde en høj oppetid. Det er også en risiko for TV 2, hvis datakvaliteten bliver lav. Dette kan fx ske, hvis der er duplikater af den samme person. Hvis datakvaliteten bliver lav, da udfylder systemet ikke den rolle det er tiltænkt, fordi de personer der deltager i produktionerne også skal have penge, ud fra deres krediteringer.

Forebygning af risici Gruppen vil benytte sig af teorien fra kurset videregående objektorienteret programmering med henblik på at skabe et system, der anvender en 3-lags arkitektur. 3-lags arkitekturen er en meget benyttet arkitektur i industrien, og sikrer at der kun er ét indgangspunkt i systemet og kan derved hjælpe til at undgå duplikeret data. Det er også med til at sikre en mindre kludret kode, hvilket gør det nemmere at vedligeholde og derved vil TV 2 nemmere kunne bevare oppetiden på deres forventede 99,7 - 99,8%.

4.3 Omkostninger og tidsplan

Fejlvurderinger i tidsestimater Hvis gruppen fejlvurderer, hvor lang tid det vil tage at implementere en bestemt feature, og således at vi overskrider tidsplanen og ikke kommer i mål med produktet som helhed.

For mange opgaver på samme tid Der kan opstå en risiko for projektets succes i slutfasen, hvis for mange opgaver er sat i gang på samme tid. Dette kan føre til uforudsete problemer ved merging af koden til Github og bør undgås i alle situationer, men især tæt på afleveringsfasen.

Forebygning af risici Gruppen lægger stor vægt på projektet og har afsat tid hver uge til at udarbejde det. Det betyder også at gruppemedlemmer står klar til at hjælpe, hvis et medlem tager en for stor opgave eller en der ligger udenfor hans evner. Gennem pair-programming kan vi sikre, at opgaver bliver løst og, at dem der har svært ved opgaven også får lært det emne, som de har svært ved. Ved brug af scum kan vi også sikre at opgaver bliver af en passende størrelse, som kan hjælpe ved merging i git, da der er mindre chance for conflicts.

4.4 Proces og værktøjer

Uerfarne gruppemedlemmer Det kan være en risiko for projektets færdiggørelse til afleveringsdato, at "svage" gruppemedlemmer får tildelt essentielle opgaver hen mod slutningen. Hvis det uerfarne gruppemedlem ikke informerer gruppen om, at han ikke kan løse opgaven, så kan det komme til at være et sidste øjebliks problem. Dette sætter pres på hele gruppen for at få lavet denne essentielle opgave så hurtigt som muligt, hvilket kan forårsage en dårlig løsning. Hvor der eventuelt kunne have været en god løsning, hvis gruppemedlemmet havde meldt det klart ud til gruppen, eller opgaven var tildelt til en anden.

Ukorrekt anvendelse af værktøjer Hvis værktøjer så som GitHub anvendes forkert, kan det være med at til at sløve gruppens arbejde, fx. hvis gruppen i fællesskab skal bruge lang tid på at fikse merge conflicts. Disse kan opstå, hvis der ikke bliver committet løbende og større mængder kode hober sig op.

Forebygning af risici I løbet af projektet vil det blive tydeligt, hvem der laver deres opgaver og hvor de forskellige medlemmer har deres styrker. Ved at kigge på dette, og holde gruppens belbin roller i mente, kan der hurtigt besluttes om et gruppemedlem kan sørge for at løse den stillede opgave, eller om han melder ud, hvis der er problemer. Når der anvendes værktøjer ukorrekt, skal alle gruppemedlemmer være opmærksomme på det, hvis det ikke kan fikses af en person, må det frem i lyset. Henover projektets forløb vil der også blive skabt kendskab til, hvordan værktøjer bruges.

5 Prioritering

I dette afsnit vil vi liste samtlige funktionelle og ikke-funktionelle krav op, efter prioriteringsanalysen MoSCoW. Ved anvendelsen af MoSCoW analysen deles kravene op i 4 grupper "MUST", "SHOULD", "COULD" og "WOULD".

5.1 Funktionelle Krav

ID	Funktionelle Krav	Beskrivelse	MoSCoW
F 01	Registrering af kreditering	Det skal være muligt for producere og firmaer at tilføje programmer og kreditere medvirkende hertil.	M
F 02	Bruger system	Prototypen af forbrugersystemet skal gøre det muligt for forbrugeren at få informationer om krediteringer og produktioner.	M
F 03	Adgangskontrol	Der skal implementeres en form for adgangskontrol, for hvem der skal have adgang til redigering, oprette og slette data.	M
F 04	Søgning på kreditering	Det skal være muligt at slå en person op og derved se samtlige programmer en person har været med i.	M

F 05	Oprettelse af personer og programmer	Der skal implementeres en skabelon, som skal udfyldes, når der skal oprettes en person eller et program.	M
F 06	Anvendelse af data	Det bør være muligt for TV 2 at eksportere en specifik mængde af data i forskellige formater som XML og CSV, så TV 2 nemt kan anvende dette data på anden vis.	S
F 07	Nem tilgang til kreditering	Efter endt show vises en kode på skærmen, som kan slås op i programmet og krediteringene til det pågældende program vises	S
F 08	Mulighed for at skelne mellem personer og firmaer	Hver person og firma har sit eget unikke ID, hvis det er for svært at skelne på ID, kan der forsøges at hente yderligere oplysninger som f.eks. telefonnummer, kaldenavn, alder.	S
F 09	Fleksibel søgning	Der behøves ikke vælges, hvad en bruger søger efter, men programmet kan matche inputtet med alle mulige entries (uanset om det er film eller person). Brugeren skal også have mulighed for eksplicit at vælge hvilken type (person, program osv.) vedkommende søger efter, evt. via en dropdown menu, således at f.eks. kun personer vises hvis det er det der vælges.	S
ID	**Funktionelle Krav**	**Beskrivelse**	**MoSCoW**
F 10	Rapportering	Det bør være muligt for TV 2 at eksportere en specifik mængde af data i forskellige formater som XML og CSV, så TV 2 nemt kan anvende dette data på anden vis.	C
F 11	Valg af sprog	Det bør være muligt at skifte mellem sprog, som minimum dansk og engelsk.	C
F 12	Notifikationer	Hver gang der sker noget i databasen skal TV 2's medarbejder modtage en notifikation med ændringer.	W

Table 7: Funktionelle krav - prioriteringer i MoSCoW

5.2 Ikke-funktionelle krav

ID	Ikke-Funktionelle Krav	Beskrivelse	MoSCoW
IF 01	Regler for kreditering	Det er vigtigt at programmet kommer til at overholde de lovmæssige retningslinjer for krediteringer	M
IF 02	3-lags arkitektur	Systemet skal bygges op som 3-lags arkitektur.	M
IF 03	Logning af redigering	Alt ændring i databasen skal logges og gemmes	M
IF 04	Oppe tid	Det skal være muligt at kunne have adgang til systemet op til 99.7 til 99.8% af tiden.	M
IF 05	Unikke personer	Det skal være muligt at linke/merge personer der refererer til den samme person i virkeligheden. Når to forskellige personer vil lave en kreditering for et program, skal den kreditering være associeret med en person og dens rolle.	S
IF 06	Sikkerhed i systemet	Systemet skal have en sikkerhed på grund af de login systemer der skal laves, og med hvem, der har adgang til hvad.	S
IF 07	Hastighed søgning og rapportering	Alt søgning og rapportering skal foregå inden 5 sekunder	C
IF 08	Integration med andre systemer	Muligheden for at interagere med andre systemer, så som TV 2's interne systemer bl.a TV 2 PLAY, men også tredjeparts systemer som YouSee, Boxer og Stofa mm	W

Table 8: Ikke-Funktionelle krav - prioritering i MoSCoW

5.3 FURPS+

Vi kategoriserer krav og hvad de vedrører med FURPS+ modellen.

5.3.1 FURPS

Functionality

ID	Navn
F 01	Registrering af kreditering
F 02	Brugersystem
F 03	Adgangskontrol
F 04	Søgning på kreditering
F 05	Oprettelse af personer og programmer
F 06	Anvendelse af data
F 07	Nem tilgang til kreditering
F 08	Mulighed for at skelne mellem personer og firmaer
F 09	Fleksibel søgning
F 10	Rapportering
F 11	Valg af sprog
F 12	Notifikationer
IF 03	Logning af redigeringer

Table 9: FURPS+ - Functionality

Usability

På grund af at dette felt er meget UX baseret har gruppen valgt ikke at sætte nogle krav til dette. Der er ikke blevet dannet nogle designmæssige krav på dette tidspunkt.

Reliability

ID	Navn
IF 04	Oppetid
IF 03	Sikkerhed i systemet

Table 10: FURPS+ - Reliability

Performance

ID	Navn
IF 08	Hastighed af søgning og rapportering

Table 11: FURPS+ - Performance

Supportatility

Gruppen har ikke nogle krav for supportability.

5.3.2 Plus

Design constraints

ID	Navn
IF 02	3-Lags arkitektur

Table 12: FURPS+ - Design constraints

Implementation requirements

ID	Navn
IF 05	Unikke personer

Table 13: FURPS+ - Implementation requirements

Interface requirements

ID	Navn
IF 09	Integration med andre systemer

Table 14: FURPS+ - Interface requirements

Hardware requirements Gruppen har ikke nogle specifikke hardware krav at tage højde for.

6 Metode i elaborationsfasen

I løbet af dette projekt ville der både blive brugt UP og SCRUM. Når vi kommer i gang med elaborationsfasen, vil gruppen gøre brug af SCRUMs sprint metode. Siden at Unified Process er delt ind i faser/iterationer er det nærliggende at bruge SCRUM til at strukturere disse iterationer. Da en iteration i UP skal give et "releaseable" produkt, ligger det sig op af et sprint i SCRUM, hvor man efter en sprint også skal have et "releaseable" produkt. Her kommer vi til at have to iterationer der strækker sig igennem elaborationsfasen. Så derfor er én iteration ét sprint i en SCRUM-proces.

Der er i gruppen bred enighed om at bruge SCRUM til at gennemføre elaborationsfasen på baggrund af at det giver stor mulighed for at arbejde effektivt og struktureret i gruppen. Da vi vil anvende SCRUM skal vi arbejde ud fra en backlog af opgaver der skal løses til hvert sprint (iteration i UP).

Det vil sige at vi går igennem UPs faser (beskrevet i afsnit 2.4), altså inceptionsfasen, elaborationsfasen, konstruktionsfasen og transitionsfasen. Under elaborationsfasen og en fremtidig konstruktionsfase vil vi gøre brug af SCRUM.

7 Ressourcer

I dette afsnit vil vi beskrive de ressourcer vi kommer til at bruge i det fremtidige arbejde med projektet. Som listet op i gruppekontrakten, har vi nogle faste arbejdsdage som er følgende:

- Der arbejdes som udgangspunkt i de 8 timer, der er afsat i skemaet hver uge.
- Derudover forventes det af hvert gruppemedlem at benytte 4-8 timer i løbet af ugen på individuelt at arbejde på projektet.

Det vil sige at vi som gruppe opnår en ugentlig arbejdsindsats på ca. 72-96 arbejdstimer, og individuel arbejdsindsats på ca. 250 timer gennem hele semestret, hvilket lever op til kravet på 10 ECTS.

8 Konklusion

Som grundlag for det problem vi er blevet stillet overfor, i forbindelse med TV2's rulletekster, har vi som gruppe startet med at indskærpe nogle krav for hvordan et system kunne se ud, for at det kan fungere som en ordentlig substitut for TV2's rulletekster.

Vi satte os ned for at undersøge problemet, og fordelene ved at have et system der krediterer de medvirkende i programmerne, frem for at have rulletekster kørende i slutningen af programmet. Vi kom så frem til en problemformulering der lyder således; Hvordan kan vi udvikle en prototype til et krediteringssystem der vil kunne erstatte de klassiske rulletekster efter et afsluttet program? Denne problemformulering er derved grundstenen i vores videre arbejde med at udvikle en kravspecifikation til dette krediteringssystem.

På baggrund af vores aktørliste, og brugsmønster-model, fik vi udarbejdet et brugsmønster-diagram, som skal agere plan for vores system. Diagrammet viser at vi har forskellige aktører der interagerer med en række brugsmøsntre på hver deres måde. Dette giver et billede af systemets flow, og er en visualisering af hvordan systemet hænger sammen i sin helhed.

Da vi nu har en klar plan for hvad systemet skal indeholde, og hvordan det skal fungere, er det vigtigt for gruppen, at vi får prioriteret vores arbejde, så vi kan få de vigtigste ting implementeret i systemet, uden at spilde tid på ligegyldige funktioner.
Af den årsag har vi udarbejdet en kravspecifikation, og opdelt kravene ved hjælp af Moscow-modellen. Dette giver os en perfekt idé af, hvad vi skal fokusere på i vores arbejde med denne case, både når det kommer til vores funktionelle krav, og vores ikke funktionelle krav. Prioriteringen over vores krav kan findes fra side 17 til side 18.

Afslutningsvis, kan vi konkludere på vores inceptionsfase, at vi har udarbejdet et klart og gennemskueligt prioriteringssystem over de funktioner der skal implementeres i det system vi vil skabe en prototype af. Derudover har vi optegnet en klar plan over vores systems struktur gennem vores brug af aktørliste og brugsmønster-diagram. Dette skaber et godt fundament, som vil give os de bedste forudsætninger for at få udviklet et system der virker efter planen.

9 Processdokumenter

9.1 Gruppekontrakt

1. Hvor ofte gruppen vil mødes:
    - Mødes 2 gange om ugen (tirsdag & fredag). Typisk én gang for at køre status og uddelegere opgaver eller lignende og én gang som vejledermøde samt hvis der er spørgsmål til en opgave eller andet vedrørende projektet.
2. Hvorvidt I tillader brugen af sociale medier under projektgruppemøder?
    - Bruges i et begrænset omfang. Så længe der er fokus på opgaven og gruppemedlemmerne.
3. Om I forventer, at alle i gruppen deltager i den undervisning, der er vigtigt for projektet?
    - Det forventes at personerne i gruppen deltager i den undervisning, der skal bruges i projektet.
4. Hvordan og hvornår en person i jeres gruppe skal fortælle, at han/hun er forhindret i at deltage en gruppeaktivitet?
    - Umiddelbart så tidligt som muligt, men helst 48 timer før. Det skal meddeles på Messenger eller Discord
5. Hvordan I skal ytre utilfredsheder, og hvordan utilfredsheder skal behandles i jeres gruppe?
    - Det er vigtigt at sige til, hvis der er noget man er utilfreds med. Det er vigtigt, at det kommer frem. Hvis der er en utilfredshed kan man tage det individuelt med personen det handler om. Vær specifik og professionel og arbejd mod en konstruktiv løsning på problemet frem for personlige anklager. Husk her at alle personer arbejde forskelligt og at den løsning, der ville virke på dig ikke nødvendigvis er den samme for vedkommende.
6. Hvordan I vil håndtere konflikter?
    - Generelt samme som 5. Ellers hvis det er helt galt, tag det åbent i gruppen eller tage det op med vejleder.
7. Jeres ugentlige arbejdstid?
    - Absolut minimum: De afsatte timer til projektet, hele tirsdagen og 8-12 fredag.
    - Derudover forventes det, at man bruger tiden udover til at lave de uddelegerede opgaver inden næste møde.
8. Jeres kommunikationsmidler og kommunikationsadfærd
(IT-baseret kommunikation)?
    - Primært discord
    - Sekundær messenger
    - Tertiær teams

9. Brugen af dagsordener og referater?

    - Vi udarbejder en dagsorden for hvert møde i starten af mødet, således at der er klarhed over, hvad der skal opnås på mødet. Denne dagsorden dokumenteres i Github Wiki. Punkter der skal med i dagsordenen:
        - Eventuelle fraværende.
        - Opdater logbog: Hvad er blevet lavet siden sidst
        - Gennemgang af, hvad der er blevet lavet siden sidst (forklaring af kode eller hvad der er blevet skrevet)
        - Eventuelle spørgsmål til vejleder.
    - Ønsker man punkter på dagsordenen til et møde, kan disse skrives i messenger før mødet, og så tilføjes den på dagsordenen.
    - Referat i forhold til gruppelog (Github Wiki)

10. Hvorvidt hjemmearbejde er tilladt.

    - Hjemmearbejde er forventet og påkrævet, i relation til de uddelegerede opgaver.
    - Hvis man udover dette får en ide, kan dette sagtens laves, men dokumenter, hvad der er blevet lavet, og tag det med til næste møde. Sørg her for ikke at gå for meget ind over et andet gruppemedlems scope. Som minimum notificer vedkommende.

11. Håndtering af nogle gruppemedlemmers studiejob og situationer, hvor det er svært at finde tidspunkter i kalenderen, hvor alle er ledige?

    - Vi forventer at alle gruppemedlemmer er ledige på de afsatte timer. Hvis det aftales at mødes udover de afsatte timer og én eller flere ikke har mulighed for at deltage, så dokumenteres arbejdet og bliver gennemgået på næste gruppemøde.

12. Konsekvenser for brud på aftaler, konsekvenser for gentagne brud på aftaler, og hvad der skal til for at et gruppemedlem på professionel vis sparkes ud af gruppen.

    - Flaskepost
    - Der skyldes en (hjemme)DATO!
    - Hvis et gruppemedlem skal sparkes ud, sker dette gennem vejleder.

13. Hvordan i organiserer jer (hvem er leder, hvem er planlægger, etc.)

    - Peter Ratgen
        - Formidler
        - Afslutter
        - Specialist
    - Casper McGuire Stillinge
        - Specialist
        - Organisator
        - Afslutter
    - Hans Rosenfeldt Pedersen
        - Afslutter

  - Analysator
  - Specialist
- Jesper Bork Diederichsen
  - Formidler
  - Kontaktskaber
  - Koordinator
- Victor Frank
  - Formidler
  - Kontaktskaber
  - Afslutter
- Jonas Beltoft
  - Afslutter
  - Specialist
  - Idémand

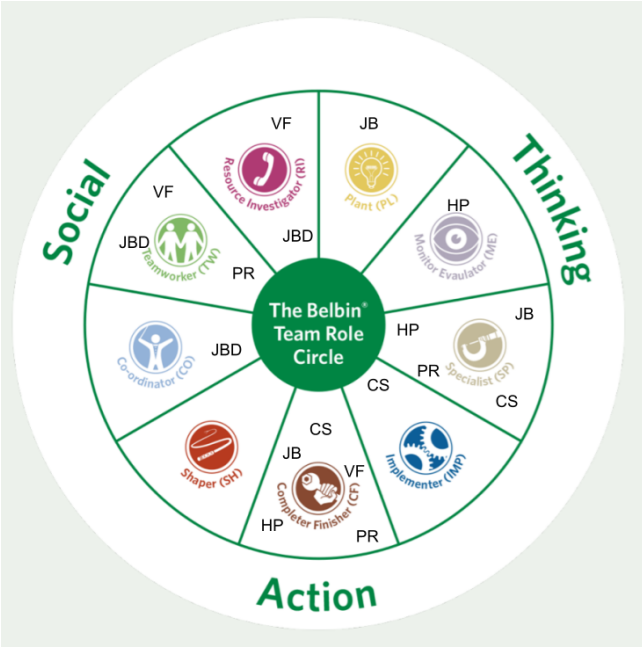


Figure 4: Simpel oversigt over vores Belbin-rolle fordeling

14. Hvordan I kvalitetssikrer de enkelte medlemmers arbejde, og hvordan I sikrer den røde tråd i hele projektet.

    - Løbende gennemgår hinandens arbejde i henhold til dagsordenen.

15. Hvordan og hvornår gruppemedlemmer, der ikke kan nå at afslutte aftalte opgaver, skal melde ud og indhente hjælp fra gruppen som helhed

    - Giv det et skud og giv dig tid, men hvis det kikser, så skriv i hvert fald 24 timer før, så nogle gruppemedlemmer har tid og mulighed for at hjælpe dig.

9.2 Vejleder kontrakt

- Referat efter alle møder.
- Alle gruppemedlemmer skal være CC'et i en mail.
- Henrik skal svare pr. mail inden for 2 hverdage.
- Henrik skal modtage en dagsorden og mødeindkaldelse til tirsdag om fredagen, og senest mandag morgen.
- Henrik skal modtage afbud til vejledermøder pr. mail.

9.3 Belbin gruppeprofil

Dette er en viderebyggelse på figur 4 på side 24 fra vores Gruppekontrakt

9.3.1 Team Styrker

- 5/6 af gruppen er afsluttere - Ergo, vi er ekstremt gode til at afslutte en opgave præcist, så længe der er lagt en klar plan.
- 4/6 i gruppen er specialister, der kan bidrage med at være selvstartende, samt kan bidrage med viden og færdigheder.
- 3/6 i gruppen er formidlere, der kan være med til at skabe et godt samarbejde i gruppemedlemmerne i mellem.
- 2/6 i gruppen er kontaktskabere, der kan være med til at skabe kommunikation i gruppen, samt med deres udadvendthed og entusiasme kan give god inspiration i gruppen.
- Godt fokus på de sociale aspekter i gruppen, hvor der vægtes et godt samarbejde, som alle trives i.

9.3.2 Team Svagheder

- Der er ikke nogle opstartere i gruppen. Det kommer til at være svært for gruppen at komme ordentligt i gang hvis der ikke er nogen der påtager sig en lederrolle
- Idet så stor en del af os er afsluttere, er der mangel på koordinatorer (samt opstarter) til at klarlægge arbejdet og gøre det nemt at komme i mål med en opgave
- Havde gruppen haft en idemand mere, kunne de to idemænd havde sparret og dermed løftet hinandens ideer. En analysator kan også sparre med en idemand, men er mere kritisk og kan skyde ideer ned.
- Grundet den store mængde af specialister og afsluttere, kan vi hurtigt komme til at tage fokus væk fra hovedopgaven for at fokusere på mindre detaljer der har lav effekt på projektet som helhed.
- Det kunne også havde været godt for gruppen at have en organisator mere, således at der kunne komme bedre styr over de mange afsluttere der er i gruppen. Også således at det sociale ikke tager overhånden over projektarbejdet.

A Bilag

A.1 Logbog

Gruppens Logbog kan findes på gruppens Github Wiki side
`https://github.com/hannehue/Semesterprojekt-2/wiki/Gruppelog`

A.2 Referat fra review

Valg af referent, Peter. Casper er ordstyrer.

A.2.1 Generelle kommentarer til inceptionsdokumentet

Joachim: Overordnet set er det et okay dokument vi har lavet. Han har en lidt anden opfattelse af problematikken omkring casen. Han syntes at der mangler en beskrivelse af hver sektion og hvad denne skal indeholde

Victor Røer: Han syntes at det er okay dokument, han har nogle specifikke formuleringer. Noget af informationen der på titelbladet, skal også fremgå på forsiden.

- Det mangler en overordnet beskrivelse af supplerende krav, i den overordnede beskrivelse af krav.
- Der skal være konkrete plan for at afværge de ricisi vi har lavet.
- Der skal være en forklaring af de prioriteringer vi har lavet (i vores MoSCoW).

Victor Andreas: Generelt et godt dokument. Der er nogle ting der mangler fra tjeklister. Der nogle sætninger der er blevet skrevet om, hvor der mangler at bliver slettet noget.

A.2.2 Specifikke kommentarer til dokumentet.

Forside Der skal være de samme informationer på forsiden som titelblad, dog flere på titelblad. Vi skal bare gå ud fra checklisten for inceptionsdokumentet.

Titelblad Det er okay.

Indledning Essensen i Joachims øjne er hovedproblemet at lette arbejdsbyrden fra TV 2. Således at TV 2 ikke skal bruge tid på at skrive krediteringerne ind manuelt, således at TV 2 kunne skal kvalitetsikre.

Problemanalyse Casen er fra TV 2 og SDU. Formålet er også at finde ud af om der egentlig er et business case for dette projekt. Så inceptionsdokument skal danne et grundlag for det er noget man skal gå videre med eller ej. I problemanalyse kunne det være godt at referere til de mange regler der er for krediteringer.

Problemformulering Igen det fra indledningen omkring, hvad casen handler om. Den sidste sætning er en smule kludret.

Fagligt videensgrundlag Victor Røer mener at vi har blandet vidensgrundlaget sammen med hvor vi har tænkt os at bruge videngrundlaget. Dette skulle måske være i et metodeafsnit.

I OOP-afsnittet omkring abstraktion, skal også fokusere på dev.

UML kan opdeles i to sektioner.

Vi mangler lidt grundlæggende viden omkring SCRUM. Vi referer til at rundt omkring i dokumentet. Det kunne også være relevant at beskrive FURPS+ og MoSCoW, i stedet for at beskrive FURPS+ i sektion 5.3.

Det fungerer godt med beskrivelsen af eksisterende løsninger.

Overordnet kravspecifikation Vi mangler at rette ud i det, omkring at brugen kan logge ind eller ej. Der var snak om at mødet med TV 2 at de kunne bruge data fra krediteringssystemet i TV 2 Play.

Victor Røer undrer sig over omkring aktøreren Informationssystem.

Producer skal formentlig oversættes til producent.

Brugsmønstre Det er være rart med en lille indledende tekst til brugsmønstre samt en egentlig overskrift til brugsmønstre. Omkring B05, var det ikke mere til firmaer så som TV 2, eller andre firmaer.

Brugsmønsterdiagram Brugmønsterdiagrammet er korrekt overordnet. Det kunne være fint med nogle bokse til de forskellige aktører.

Detaljerede brugsmønstre Problematik omkring punkt 5 i B03, skal TV 2 selv sidde og tage stilling omkring skabelon. Er det bare godkendelse, eller skal TV 2 sidde og skriv ind.

Kritiske risici Afværgelse af risici.

Prioritering

MoSCoW Forklaringer af prioriteringer.

FURPS+ Sammenblanding af sprog i overskrifter.

Ressourcer Tidsplanlægning Fredag.

Konklusion LaTeX fejl i andensidste afsnit. Den opsummerer måske lidt for meget.

Bilag Logbogen skal indsættes. Checklisten skal indsættes.

B Checkliste for inceptionsdokument

Indhold	Kriterier for godkendelse	Opfyldt +/-/i
Forside	Projekttitel, uddannelsesinstitution, fakultet, institut, uddannelse, semester, kursuskode, projektperiode, vejleder, projektgruppe og projektdeltagere (fornavn, efternavn, sdu-email). Må gerne have illustrationer.	+
Titelblad	Samme oplysninger som på forsiden, samt projektdeltagernes underskrifter (Projektdeltagernes aktive deltagelse i projektforløbet anerkendes gensidigt ved projektdeltagernes underskrifter. Må ikke have illustrationer.	+
Indholdsfortegnelse	Samlet indholdsfortegnelse for hele projektrapporten. Højst to eller tre niveauer i indholdsfortegnelse, der kan evt. være flere i selve rapporten. Afsnit på niveau 1 og 2 skal være nummererede.	+
Indledning	Projektets rammer og baggrunden for projektet. Formål med og mål for inceptionsfasen. Problemanalysen. Problemformulering og afgrænsninger.	+
Faglige vidensgrundlag	Begrebsdefinitioner, teori og fagligt vidensgrundlag. Relevante eksisterende løsninger.	+
Hovedtekst		
Overordnet kravspecifikation	Overordnet kravspecifikation med, Overordnet brugsmønstermodel: Afgrænsning, brugsmønsterdiagram, aktørbeskrivelser, korte brugsmønsterbeskrivelser, samt detaljeret beskrivelse af udvalgte, essentielle brugsmønstre, Overordnet beskrivelse af supplerende krav	+
Kritiske risici	Identificerede kritiske risici, dvs. risici som vil være en trussel mod projektets succes, hvis de ikke bliver afværget: Er alle kritiske risici identificeret? Er de identificerede risici blevet afværget, eller er der en plan for at afværge dem?	+
Prioritering	Prioritering der bygger på kravenes forretningsmæssige betydning (nytte/benefit), arkitektoniske betydning, risiko samt læringsmæssige udbytte.	+
Metode i elaborationsfasen	Metode, der vil blive brugt i elaborationsfasen, herunder en overordnet beskrivelse af den måde som UP og Scrum vil blive kombineret i projektet.	+
Resurser	Arbejdstidsindsats uge for uge og alt i alt	+
Konklusion	Opsummering af resultaterne og diskussionen af dem. Svar på om formålet og målene med inceptionsfasen er opnået.	+
Procesdokumenter	Gruppekontrakt, opdateret, Vejlederkontrakt opdateret, Belbin gruppeprofil	+
Bilag		
Logbog	Link til logbogen	+

Indhold	Kriterier for godkendelse	Opfyldt +/-/i
Review	Referat af review	+
Checkliste for inceptionsdokument	Udfyldt checkliste	+

References

[Sej] Rene Sejberg. *Introduktion til Scrum*. URL: https://danskprojektledelse.dk/community/introduktion-til-scrum/. (tilgået: 18/03 - 2021).

[Som15] Ian Sommerville. *Software Engineering*. Pearson, 2015. ISBN: 0133943038.