Forsiden

Anders Looft, Thomas Nielsen,

Simon Lorentsen og Lasse Meilby.

//he15dmu-2s14

// FFS

//Førsteårsprøven

//Erhvervsakademi MidtVest

//Datamatiker

//Dato Juni 2015

//Vejleder: Anders W. Petersen, Flemming K. Jensen, Hans Iversen

Indholdsfortegnelse

[Opgave start 1](#_Toc420401938)

[Indledning - MANGLER 1](#_Toc420401939)

[BPR 2](#_Toc420401940)

[UCD 2](#_Toc420401941)

[Reverse Engineering 2](#_Toc420401942)

[Forward Engineering 3](#_Toc420401943)

[Visionsdokument 3](#_Toc420401944)

[Vision: 3](#_Toc420401945)

[Interessentanalyse: 4](#_Toc420401946)

[Featureliste: 4](#_Toc420401947)

[Use case Diagram 5](#_Toc420401948)

[Domænemodel 6](#_Toc420401949)

[Datamodel 7](#_Toc420401950)

[Endelige overvejelser 8](#_Toc420401951)

[Systemsekvensdiagram 10](#_Toc420401952)

[Operationskontrakter 10](#_Toc420401953)

SKRIV BILAG HER (opdateres ikke, pga. sidetal)

# Opgave start

# Indledning - MANGLER

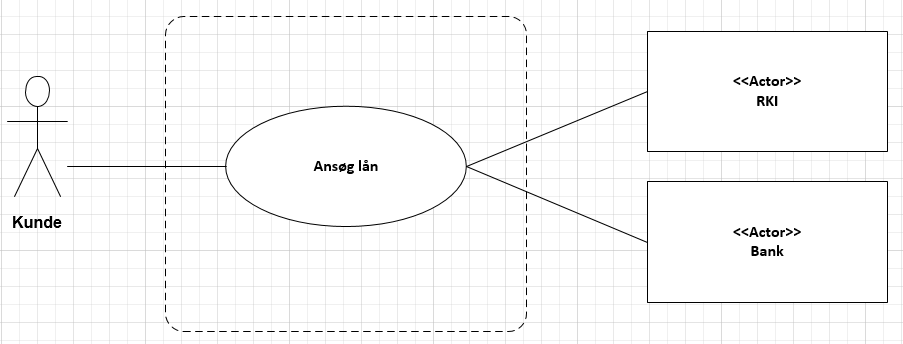
Vi har i dette projekt anvendt reverse og forward engineering, for at kunne beskrive det nuværende system, og det fremtidige system. Projektet er blevet gennemført ved anvendelse af Unified Process, og dokumenteret med UML notation.

Programmet er kodet i Java

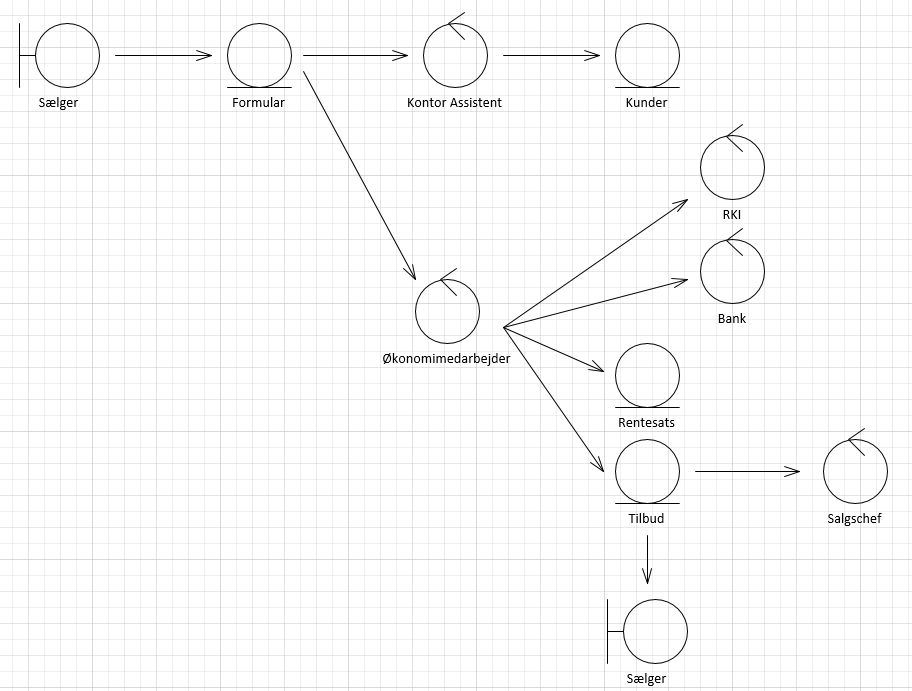
# BPR

## UCD

Vi har her lavet et Use Case Diagram for virksomhedens perspektiv. Dette giver os en idé om, at RKI og Bank spiller en stor rolle, når en kunde skal ansøge om et lån. UCD for både Reverse og Forward ser således ud, da dette ikke burde ændres ved implementationen af vores nye system.



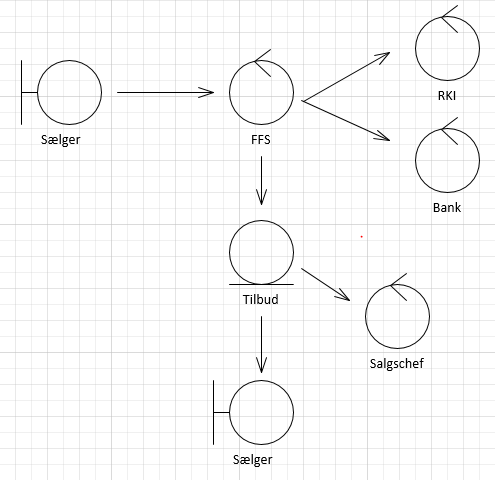
## Reverse Engineering

Vi har ved hjælp af Reverse Engineering, kunne identificere problem domænet, hvilket har givet os en forståelse for, hvordan det fremtidige system skal se ud. Object modellen viser at Økonomimedarbejderen har ekstrem meget ansvar i det nuværende system, da Økonomimedarbejderen skal igennem en masse processer før han/hun kan give lånetilbuddet videre til sælgeren, eller salgschefen, afhængig af lånetilbuddets størrelse, og om kunden har voldt problemer før.

## Forward Engineering

Vi har her ved hjælp af Forward Engineering, kunne fjerne store problemer ved det nuværende system. Objekt modellen nedenfor viser en tydelig forbedring i det fremtidige system. Dette udtrykkes ved at systemet(FFS) er kommet ind i billedet. Ved hjælp af FFS er det mulig of fjerne 2 control-objekter (Økonomimedarbejder og Kontor Assistent). Entity-objektet Formular ligger nu i FFS.

FFS står nu for at hente informationer fra RKI og bank. FFS sætter hermed også Rentesatsen, så der ikke længere er brug for en medarbejder til at gøre det. Dette systematisere processen betydeligt, hvilket realisere et hurtigere og mere effektivt system.



# Visionsdokument

## Vision:

Vi forestiller os et system der er baseret på at realisere en kundes drømme uden ventetid. Ferrari Financing System (FFS) har en automatiseret forbindelse til RKI og banken, og kan udregne det bedst mulige lånetilbud baseret på hver enkelt kunde inden for få minutter. Med FFS kan kunden altså få sin drømmebil med hjem den selv samme dag. Derudover vil FFS gøre det muligt for sælgeren at have flere salg med den automatiserede proces. Med vores system vil kunden trygt kunne udlevere sine personlige informationer, da systemet håndterer personlige informationer med diskretion.

## Interessentanalyse:

Kunden er interesseret i at hans/hendes oplysninger bliver gemt fortroligt. Samt at processen ikke tager for lang tid. Endvidere at han/hun kan få det bedst mulige tilbud.

Sælgeren er interesseret i et system der giver hurtig og korrekt respons, for at kunne lave flere salg.

Salgschefen er interesseret i at kunne modtage de lånetilbud der skal godkendes elektronisk. Endvidere er han interesseret i at forøge salget, og sikre høj kvalitet ved udregning af tilbud.

RKI har interesse i at personlige data bliver behandlet med diskretion, og sikkerheden til deres system er vedligeholdt.

Banken har interesse i at processen forløber hurtigt, og sikkerheden til deres system er vedligeholdt.

Ejeren har interesse i at der kan laves lånetilbud på kort tid, med høj kvalitet. Dette kan øge salget.

Datatilsynet har interesse i at personfølsomme oplysninger behandles korrekt og sikkert, i forhold til de gældende regler og love.

## Featureliste:

Lånetilbud

* Oprettelse
* Kreditvurdering
* Udregning af rentesats
* Godkendelse
* Eksportering
* CSV-fil

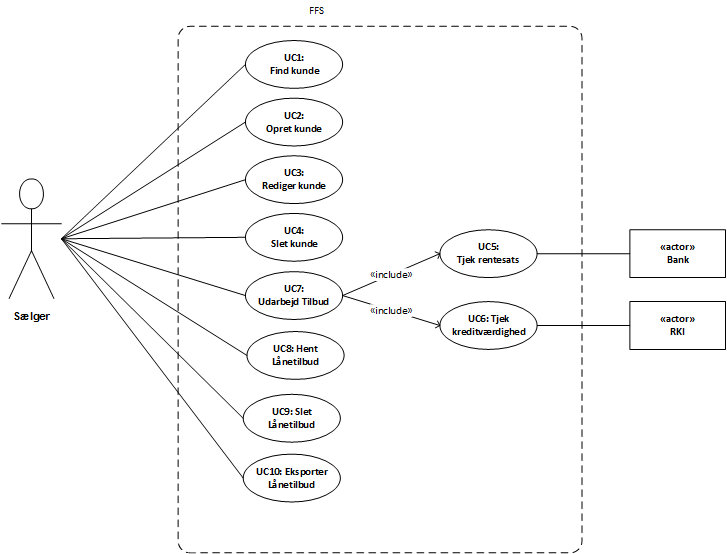
Kundehåndtering

Persistering af data

OM visions dokumentet.

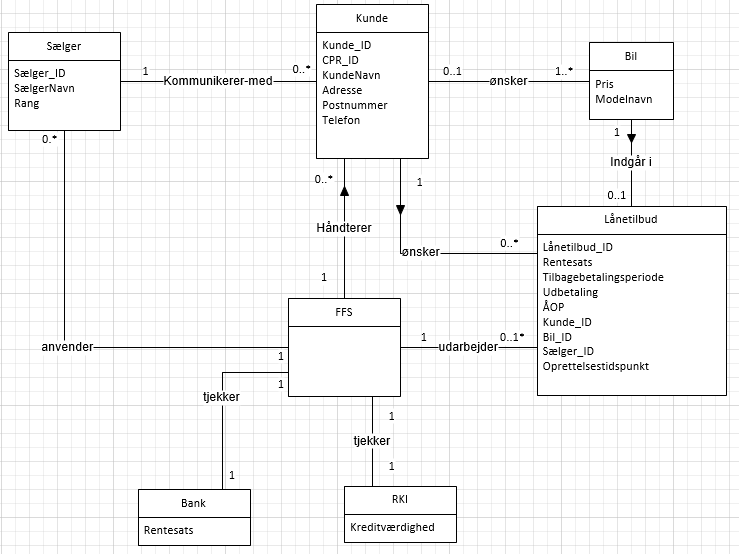
Visions dokumentet er et dokument som indeholder en vision, selve visions teksten, er beskrivelsen som skal sælge systemet til kunde, og overbevise kunden om at systemet er uundværligt, så derfor er det vigtigt at fokuserer på at visionen ikke er en kedelig opremsning. Ud over visions teksten indeholder dokumentet også en interessent analyse, som nævner hvem der kan have interesse i systemet og hvad de har interesse i. til Sidst er der en featureliste, som viser funktionerne i systemet i en rækkefølge med de vigtigste features øverst.

# Use case Diagram

Et use case diagram (UCD) er en artefakt, som man anvender til at få et overblik over alle use cases (UC) som er i systemet. Ligeledes giver det et overblik over hvem der har med de forskellige use cases at gøre altså aktørerne. Der findes både primære – og sekundære aktører. Den primære aktør vises altid til venstre i et UCD, og er den som er anvender den enkelte use case. Den sekundære aktør vises altid til højre i et UCD og kan for eksempel være et udefrakommende system som ens eget system interagerer med for at kunne udfører en given use case.   
i et use case diagram arbejdes der med 2 forskellige slags use cases, konkrete - og abstrakte use cases, hvor den konkrete er den som bliver startet af en aktør, og den abstrakte som startes af en anden use case. 

I vores UCD over FFS har vi en primær aktør som er sælger, fordi det er ham der anvender vores system og ham der interagerer med hver enkelt use case, som kan ses ved at der er en streg fra sælger til hver enkelt use case. Alle de use cases som er i forbindelse med sælgeren er derfor konkrete use cases og de to sidste UC5 og UC6 er abstrakte use cases. Udover at være en abstrakt use case har UC5 også en sekundær aktør RKI if UC6 har Bank som sekundær aktør.

# Domænemodel



Domænemodellen for det samlede system, har vi lavet for og give et klart overblik over problemdomænet. Vi ser tydeligt at FFS skal håntere, udarbejde og anvende en masse forskellige ting. Domænemodellen giver os her et overblik over hvordan strukturen vil komme til og se ud. Og hvordan et lånetilbud egentlig bliver oprettet.



Vi har valgt og vise domænemodellen for UC7, da det er en af de centrale use cases for vores system.

I denne use-case ser vi hvordan en sælger har kontakt med 1 eller flere kunder som ønsker et lånetilbud. Sælger anvender derefter systemet (FFS), til at udarbejde et lånetilbud til kunden. Systemet har kontakt til RKI og bank som returnerer henholdsvis Kreditværdighed og Rentesats til systemet.

# Datamodel

Figur xx: Tidligt udkast af datamodel



Tidligt i forløbet så vores datamodel således ud. Dette var første udkast, og primært brugt til et overblik over hvilke værdier der skulle være i databasen. Vi udvidede hurtigt modellen til at inkludere information omkring sælger og bil, med dertilhørende værdier. Disse to blev koblet på lånetilbud med en foreign key/primary key relation. Dette tillod os at have tabeller med sælgere og biler oprettet, og blot linke dem på når et nyt lånetilbud blev lavet. Modellen levede stadig ikke op til de 3 normalformer. 1. normalform er bestået, da alle attributter kun har én værdi.

Det første problem lå i bynavn og postnummer, da bynavnet er fuldstændig afhængigt af postnummeret. Disse to blev derfor sat i en tabel for sig selv, og så lod vi postnummeret i kundetabellen referere til postnummeret i postnummer tabellen. Således havde vi altid både postnummer og bynavn tilgængeligt, såfremt vi kendte postnummeret.

Efter at vi havde flyttet pris ud i en tabel sammen med de nye attributter kunne modellen bestå 2. normalform, da alle attributter nu var fuld afhængige af den primary key der ligger i den tilsvarende tabel.

Til sidst valgte vi at lade CPRnummer ligge i en tabel for sig selv, men en dertilsvarende CPR\_id, der refererede tilbage til kunde tabellen. Det kan argumenteres at datamodellen allerede her opfyldte den 3. og sidste normalform, da der ikke direkte var nogle transitive afhængigheder. Det blev dog besluttet at flytte CPRnummeret ud, da dette evt. i en fremtidig revision kunne øge sikkerheden. Den opdaterede datamodel kan ses på figur xx nedenfor.

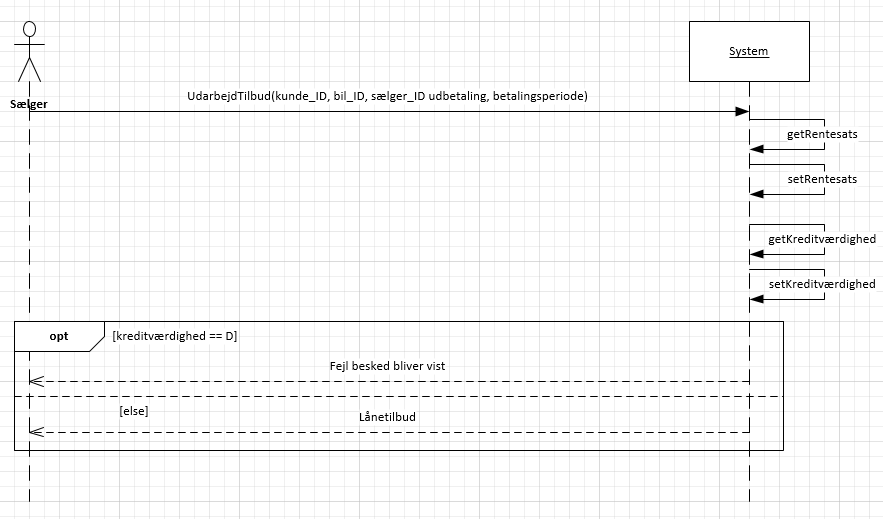
## Endelige overvejelser

Databasen har et begrænset omfang, og der er helt klart plads til forbedringer. Vi har foretaget nogle valg der måske går imod visse standarder, for at holde det simpelt. Bil tabellen indeholde kun attributterne pris og modelnavn. I et virkeligt system ville dette ikke være tilstrækkeligt, da der ikke er plads til fx tilvalg eller tilbud.

Ligeledes findes der i sælgertabellen en attribut; beløbsgrænse. Denne *burde* være i en tabel for sig selv, forbundet til sælgertabellen med primary-/foreignkey relation via rang. Vi har valgt at begrænse vores system til ganske få sælgere, og derfor tillod de enkelte gentagne værdier i samme kolonne der nu måtte være. Det betyder selvfølgelig at det ikke overholder normalformerne til fulde. 

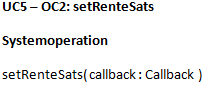
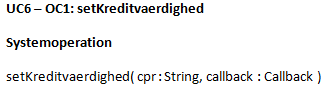
Figur xx: Datamodel

# Systemsekvensdiagram



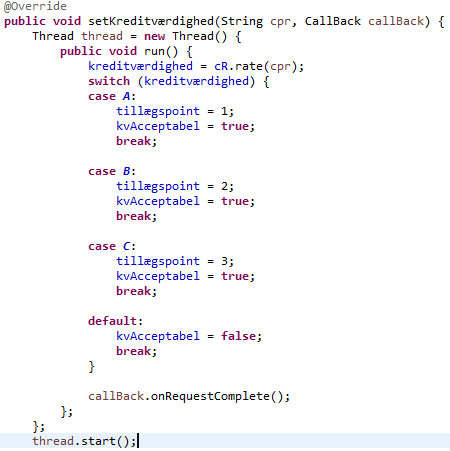
Systemsekvensdiagrammet for UC7 – Udarbejdtilbud, går i sin store helhed ud på at systemet skal have: en kunde, bil, sælger et beløb samt en betalingsperiode, for og kunne fortsætte processen. Vi har vi dette tidelige stadie klar over, at en rentesats og en kreditværdighed, skulle findes før et tilbud kunne beregnes. Metode kaldet UdarbejdTilbud sender de nødvendige parameter med, og herefter starter systemet med og finde rentesatsen den given dag, og finde kreditværdigheden på kunden, som blev sendt med som parameter. Systemet tjekker om kunden har kreditværdigheden: D. Hvilket resultere i en fejl besked, da systemet ved allerede her at kunden ikke vil få lånet godkendt. Hvis kunden har kreditværdigheden A, B eller C, fortsætter systemet med beregningerne, og sender så tilsidst et lånetilbud tilbage til sælgeren.

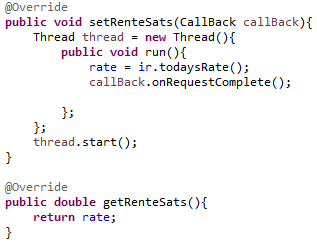
# Operationskontrakter

Ved operationskontrakterne har vi valgt og fokusere på centrale dele af systemet. Ved Usecase diagrammet blev der vist, en tydelig forbindelse imellem UC5+UC6 til UC7. I eksemplet ovenover ser vi vores første operationskontrakt: setKreditvaerdighed. Hvis vi ser på parameterne på denne metode(og får OC2), er der klart at en parameter springer i øjene: Callback.

Ideen med Callback parameteren var, at vi allerede her vist at disse to metoder(setKreditvaerdighed, setRenteSats) skulle køres i tråde. Her ser i koden til setKreditvaerdighed:

  
Callback parameteren tillader os og kalde en metode som hedder onRequestComplete(). Efter at tråden er blevet startet, bliver der oprettet forbindelse til RKI, hvilket så resulterer i en kreditværdighed. Dette bliver så tjekket igennem switch sætningen, hvor efter callback kalder onRequestComplete() på sig selv. Dette gør at systemet ved præcis hvornår metoden setKreditværdighed() er færdig. Ellers skulle systemet sidde og pinge i et fast interval, for og finde ud af hvornår den er færdig.



Ovenfor ser vi metoden setRenteSats(). Ideen er den samme som for setKrediværdighed(). Dette gør vores 2 tråde meget mere effektive, eftersom at systemet kan fokuser på andre opgaver, indtil at onRequestComplete() bliver kaldt.

Operationkontrakterne kan ses i deres helhed i bilagene.

Bilag