Danmarks Tekniske Universitet

INSTITUT FOR INFORMATIK OG MATEMATISK MODELLERING

Eksamensprojekt

Service orienteret arkitektur med Windows Communication Foundation

Kgs. Lyngby Juli 2006

Danmarks Tekniske Universitet
Institut for informatik og Matematisk Modellering

TITEL:

Service orienteret arkitektur med Windows Communication Foundation

PROJEKTPERIODE:

Eksamensprojekt 35 point,
Foråret 2006

UDFØRT AF:

Rasmus Reitz s002008
Peter Thornton-Eriksen s001967

VEJLEDERE:

Steen Eriksen NextUs A/S
Bjarne Poulsen DTU, IMM
Michael R Hansen DTU, IMM

1.2 RESUME

Rapportens formål er at undersøge anvendeligheden af en service orienteret arkitektur. Med udgangspunkt i udvikling af et kvalitetssikringssystem til kundekontaktcentre. Vil det blive undersøgt, om der med fordel kan opstilles en service orienteret arkitektur til løsning af de opstillede krav til systemet.

Efter en analyse af den opstillede specifikation på systemet, vil der med en service orienteret tilgang til problemstillingen blive opstillet en arkitektur. Det udarbejdede design vil med Windows Communication Foundation og .NET blive tilstrækkelig implementeret og testet, til at dokumentere om det kan konkluderes at der opnås umiddelbare fordele, som underbygger at, der med succes kan anvendes en service orienteret arkitektur til løsning af det ønskede system.

1.3 ABSTRACT

The purpose of the report, is to examine the usability and advantages of a service-oriented architecture. Based on development of a quality assurance system for customer service departments, the usability and advantages, of utilizing a service-oriented architecture in such a system, will be examined.

Based on analysis of the specifications for the system, and the principles of service orientation an architecture will be defined. The defined architecture will be implemented and tested using Windows Communication Foundation and .NET to a degree where be used in the examination.

1.4 FORORD

Denne rapport er udarbejdet som Civilingeniør eksamensprojekt ved Informatik og Matematisk Modellering på Danmarks Tekniske Universitet. Vejledere på projektet har været Bjarne Poulsen og Michael R. Hansen. Projektet er udført i samarbejde med NextUs A/S på baggrund af et oplæg fra Tommy Langhoff fra Kundeservice Gruppen.

Vi vil gerne takke følgende personer:

Bjarne Poulsen og Michael R. Hansen for råd og vejledning gennem hele projektet.

Steen Eriksen fra NextUs A/S for råd, vejledning og korrekturlæsning af rapporten.

NextUs A/S for at stille udstyr og lokaler til rådighed.

Tommy Langhoff fra Kundeservice Gruppen for bistand med praktisk erfaring og viden indenfor kundekontaktcentre branchen.

Ditte Jensen for korrekturlæsning.

Kgs. Lyngby, Juli 2006

Rasmus Reitz, s002008

Peter Thornton-Eriksen, s001967

KAPITEL 1 - INTRODUKTION 1

1.1 BAGGRUND 2
1.2 KUNDEKONTAKTCENTER 2
1.3 DET ØNSKEDE SYSTEM 3
1.4 PROBLEMFORMULERING 4
1.5 SERVICE ORIENTERING / SERVICE ORIENTERET ARKITEKTUR 4
1.5.1 Service orientering / Objekt orientering 4
1.6 IDENTIFIKATION AF SERVICES I SYSTEMET 5
1.7 PROJEKT AFGRÆNSNING 6
1.7.1 Tidsplan 6
1.8 RAPPORTENS STRUKTUR 7

KAPITEL 2 - KRAVSPECIFIKATION 8

2.1 KRAVSPECIFIKATION FOR SERVICES 9
2.1.1 System 9
2.1.2 Evaluering 12
2.1.3 Questionaire 13
2.2 RESUME 14
2.2.1 Systemservice 14
2.2.2 Evaluering 14
2.2.3 Questinarie 14

KAPITEL 3 - ANALYSE 15

3.1 AKTØRER 16
3.1.1 Aktør diagram 16
3.1.2 Aktør beskrivelser 16
3.2 SYSTEM SERVICE 18
3.2.1 Use case model 19
3.2.2 Domænemodel 23
3.3 EVALUERING 24
3.3.1 Use case Model 25
3.3.2 Domænemodel 26
3.4 QUESTIONAIRE 27
3.4.1 Use case Model 27
3.4.2 Domænemodel 31
3.5 RESUME 32

KAPITEL 4 – SERVICE ORIENTERET ARKITEKTUR 33

4.1 TRADITIONEL ARKITEKTUR 34
4.2 SERVICE ORIENTERET ARKITEKTUR 36
4.3 DE FIRE HOVEDREGLER 37
4.3.1 Hovedregel 1: Grænser er eksplicitte 37
4.3.2 Hovedregel 2: Services er autonome 37
4.3.3 Hovedregel 3: Services deler skema og kontrakter ikke objekter og klasser 38
4.3.4 Hovedregel 4: Kompatibilitet er baseret på policies 38
4.4 KARAKTERISTIKA VED SO/SOA 39
4.5 WINDOWS COMMUNICATION FOUNDATION (WCF) 40
4.5.1 Grundlaget for Windows communcation foundation 40
4.5.2 WCF's ABC - Address, Binding, Contract 41
4.5.3 Sikkerhed 46
4.5.4 Besked udveksling 47
4.5.5 Hosting 47
4.6 RESUME 48

KAPITEL 5 - DESIGN **49**

5.1 BAGGRUND FOR DESIGN 50
5.2 OVERORDNET ARKITEKTUR 50
5.2.1 Sikkerhedsmekanisme *51*
5.2.2 Service skabelon *52*
5.3 SYSTEMSERVICE 55
5.3.1 Kontrakter *55*
5.3.2 Klasser i system-servicen *56*
5.3.3 Tilføj service til systemet *58*
5.3.4 Database design *59*
5.4 QUESTIONAIRESERVICE 61
5.4.1 Kontrakter *61*
5.4.2 Klassediagram *62*
5.4.3 Database design *64*
5.5 EVALUERINGSERVICE 65
5.5.1 Kontrakter *65*
5.5.2 Klassediagram *67*
5.5.3 Database design *68*
5.6 RESUME 69

KAPITEL 6 - IMPLEMENTERING **70**

6.1 WCF IMPLEMETERING 71
6.1.1 Arkitektur *71*
6.2 VALIDERING/AUTORISERING I SIKKERHEDSMEKANISMEN 73
6.2.1 CiwUserNamePasswordValidator *73*
6.2.2 CiwServiceAuthorizationManager *74*
6.3 ADDSERVICE OPERATION 75
6.4 DATA ACCESS 77
6.4.1 Stored procedures *77*
6.4.2 Dataset *77*
6.5 RESUME 77

KAPITEL 7 - TEST **78**

7.1 TESTMETODER 79
7.1.1 Arkitekturtest *79*
7.1.2 Unittests *80*
7.1.3 Accepttest *83*
7.2 RESUME 83

KAPITEL 8 - KONKLUSION **84**

8.1 PROBLEMFORMULERING 85
8.2 OPSUMMERING 85
8.3 KONKLUSION 86
8.4 FREMTIDIGT ARBEJDE 87
KILDEFORTEGNELSE 88

FIGUR TABEL

Figur 1 – System skitse 5
Figur 2 – Begrænset systemmodel projekt fokusområde 6
Figur 3 – Kravsmodel for bruger administration 10
Figur 4 – Kravsmodel for gruppe administration 10
Figur 5 – Kravsmodel for rolle administration 11
Figur 6 – Kravsmodel for service administration 11
Figur 7 – Kravsmodel for Evalueringservicen 12
Figur 8 – Kravsmodel for Questionaireservicen 13
Figur 9 – Aktør diagram 16
Figur 10 – Hierarki i systemservicen 18
Figur 11 – Use case diagram til sikkerhedshåndtering 19
Figur 12 – Use case diagram brugeradministration 20
Figur 13 – Use case diagram for gruppeadministration 21
Figur 14 – Use case diagram for rolle administration 22
Figur 15 – Use case diagram service administration 23
Figur 16 – Domænemodel til systemservice 23
Figur 17 – Use case diagram Evalueringservice 25
Figur 18 – Domænemodel til evalueringsservice 26
Figur 19 – Use case diagram for Questinaire servicen 27
Figur 20 – Domænemodel for Questionaireservicen 31
Figur 21 – Traditionel arkitektur 34
Figur 22 – Service Orienteret Arkitektur 36
Figur 23 – Windows Communication Foundations ABC 41
Figur 24 – IExtensibleDataObject 44
Figur 25 – Overordnet arkitektur 50
Figur 26 – Sikkerhedsstruktur 51
Figur 27 – Service skabelon 52
Figur 28 – Klassediagram for CiwSecurity 53
Figur 29 – Klasse diagram ErrorHandling assembly 53
Figur 30 – Klassediagram for systemservicen 57
Figur 31 – System sekvens diagram for tilføjelse af en service til systemet 58
Figur 32 – Gruppe udvidelser for system database 59
Figur 33 – Service udvidelser for systemservice 60
Figur 34 – Klasse diagram for Questionaireservicen 63
Figur 35 – Database diagram for Questionaireservicen 64
Figur 36 – Klasser i Evalueringservicen 67
Figur 37 – Database Diagram Evalueringservice 68
Figur 38 – Dataset til CiwServices klassen 77
Figur 39 – Illustration af kald under arkitekturtesten 80
Figur 40 – Resultat af unittests for AddService 82
Figur 41 – Code Coverage resultat efter kørsel af unittests for AddService 82
Figur 42 – Mockup model der benyttes til AddService test 83
Figur 43 – Manuel kontrol af data i databasen vha. Microsoft SQL Server Management Studio 83

LISTE AF TABELLER

Tabel 1 – Krav til adgangskontrol i systemservicen....9
Tabel 2 – Forskal til protokol i Evalueringservicen....24
Tabel 3 – Standart binding typer i Windows Communication Foundation....45
Tabel 4 – Legitimationstyper ved forskellige transporttyper i WCF....46
Tabel 5 – Datakontrakter i systemservicen....56
Tabel 6 – Klasser i systemservicen....56
Tabel 7 - Datakontrakter vedrørende test og analyser....61
Tabel 8 – Datakontrakter vedrørende besvarelse af test og analyser....62
Tabel 9 – Beskrivelse af klasser i Questionaireservicen....62
Tabel 10 – Datakontrakter omhandlende administration af skemaer i Evalueringservicen....65
Tabel 11 – Datakontrakter omhandlende evalueringer og skemadata i Evalueringservicen....66
Tabel 12 – Datakontrakt til lederkalibrering i Evalueringservicen....66
Tabel 13 – Beskrivelse af klasserne i Evalueringservicen....67
Tabel 14 – Unittests i arkitekturtesten....79

LISTE AF EKSEMPLER

Eksempel 1 – Service kontrakt....42
Eksempel 2 – Service implementering....42
Eksempel 3 – Datakontrakt....43
Eksempel 4 – Implemtering af CiwUserNamePasswordValidator....73
Eksempel 5 – Implemtering af CiwUserAuthorizationManager....74
Eksempel 6 – AddService operation i systemservicen....75
Eksempel 7 – Testmetode til AddService funktionen....81
Eksempel 8 – Testmetode til AddService der forventer, der kastes en exception....81

Kapitel 1

INTRODUKTION

Der vil i dette kapitel blive redegjort for baggrunden for projektet, problemstillingen og hvilke områder opgaven vil begrænse sig til.

Da der i projektet arbejdes med en løsning, der skal anvendes i kundekontaktcentre, vil der desuden være en generel beskrivelse af opbygningen af kundekontaktcentre, deres virkeområde samt arbejdsrutiner. Der vil sidst i kapitlet kunne findes en kapitel-oversigt.

1.1 BAGGRUND

Projektet tager udgangspunkt i en henvendelse fra Tommy Langhoff, som gennem sit arbejde med rådgivning og uddannelse af kundekontaktcentre har specificeret et kvalitetssikringssystem til disse. Oplægget kan findes i bilag 1. Tommy Langhoff har gennem de sidste år arbejdet med ledelse og styring af kundekontaktcentre. Tommy Langhoff er i dag en del af Kundeservice Gruppen[1], som sælger konsulentbistand til kundekontaktcentre. Gennem arbejdet har Tommy Langhoff opnået viden og erfaring om kundekontaktcentres arbejdsmetoder samt hvilke problemer og svagheder, der findes i deres rutiner.

Tommy Langhoffs primære opgave er at rykke kundekontakt-centre tættere på en optimal ledelse og styring. Dette skal gavne medarbejdere, ledelse og i sidste ende kunderne. I arbejdet med forbedring af arbejdsmetoderne har Tommy Langhoff set et stigende behov for et system, der på en standardiseret og ensrettet måde kan sikre kundekontaktcenteret en stigende kvalitet i kontakten med kunderne.

Tommy Langhoff har i fællesskab med NextUs A/S indgået en aftale om udvikling af dette system. NextUs stiller i den forbindelse den tekniske ekspertise til rådighed for udvikling af systemet. Tommy Langhoff vil gennem udviklingsarbejdet bidrage med den fornødne faglige viden omkring kundekontaktcentre. Tommy Langhoff vil desuden i slutningen af processen, bidrage med kundeobjekter til test for at opnå et system, der rammer markedet bedst muligt.

NextUs, Tommy Langhoff og de studerende har i samråd opnået enighed om at udvikle en del af dette system som eksamensprojekt. Dette vil resultere i en længere udviklingsperiode, men man mener, at dette er værd at satse på for til gengæld at opnå en mere nuanceret og velovervejet tilgang til analyse og design af løsningen.

1.2 KUNDEKONTAKTCENTER

Det er i enhver virksomhed et vigtigt aspekt at pleje og supportere sine kunder. Det er vigtigt at dialogen mellem firma og kunder er professionel og ligger kvalitetsmæssigt på et højt niveau. Kundekontaktcentre er derfor i stigende grad en del af organisationen, der stjæler fokus.

Behovet for et kundekontaktcenter eksisterer i dag i alle brancher, som i kraft af deres produkter har brug for at føre en dialog med deres kunder. Dette vil som oftest være i form af salg, service eller support. Strukturen i et kundekontaktcenter er ofte hierarkisk opbygget og består af ledere, mellemledere samt et antal medarbejdere. Kundekontaktcenteret opererer som oftest med forskellige grupperinger. Disse grupperinger er anvendt til at skille medarbejdere i mindre virkeområder eller specifikke produkt grupper alt efter behov.

Den enkelte medarbejders kontakt med kunden er i næsten alle tilfælde via telefon eller e-mail/brev. Det daglige arbejde for en medarbejder er typisk at yde support inden for sit virkeområde. Udover dette tilhører medarbejderen måske en eller flere grupper med tilknytning til specifikke firmaprodukter. Medarbejderen tager imod opkald eller besvarer breve fra kunder angående fejlretning, anvendelse af produktet eller andre problematikker af lignende karakter. Samtalen registreres med hensyn til type, emne udfald, kundetilfredshed samt andre parametre, som gør sig gældende på det specifikke område.

Den anden del af et kundekontaktcenteret er leder og mellemleder staben, som beskæftiger sig med ledelse og administration af centeret. Den primære opgave er her at lave løbende kvalitetskontrol af de enkelte grupper og medarbejdere. Kvalitetskontrol og

[1] www.kundeservicegruppen.dk

udvikling/uddannelse af den enkelte medarbejder er den naturlige måde at højne kvaliteten af centeret, da det er den enkelte medarbejder, der gennemfører de få vigtige minutter i kontakten med kunden, og på denne måde er centerets ansigt udad til.

Lederne i centrene bruger derfor meget af deres tid på at foretage denne løbende kvalitetskontrol. I forbindelse med dette indsamler lederne information fra medarbejderne vedrørende kontakten med kunderne. Ud fra disse informationer udfærdiges en række rapporter der skal give ledelsen et billede af service niveauet i centeret. Herunder hvilke sager der behandles, hvad resultaterne er, samt hvad er kundernes tilfredshed. En del af ledernes arbejde er ligeledes at observere den enkelte medarbejder i hans kontakt med kunden og foretage evalueringer på dette, således at medarbejderen kan udvikles fagligt og udføre et bedre stykke arbejde.

Det er i denne sidste del af kundekontaktcenter rutinerne projektet her har sin berettigelse. Som tidligere nævnt hjælper Tommy Langhoff lederne med at lave en mere målrettet kvalitetskontrol af den enkelte kundekontakt medarbejder. Med afsæt i sine erfaringer har Tommy Langhoff beskrevet et system bestående af en række værktøjer, som hver især dækker et behov hos lederne for nemmere at gennemføre de arbejdsrutiner der knytter sig til arbejdet med kvalitetssikring i kundekontaktcenteret.

1.3 DET ØNSKEDE SYSTEM

Systemet, der ønskes udviklet, skal realisere behovet for et kvalitetssikrings system til anvendelse i kundekontaktcentre og vil ud fra Tommy Langhoffs beskrivelse skulle indeholde følgende værktøjer.

- Evalueringsskemaer – Standardiserede skemaer og protokoller til brug ved evaluering af medarbejdere. Skemaerne anvendes når lederne ud fra fastlagt procedure foretager en evaluering af medarbejderne under igangværende samtaler eller anden kontakt med kunder.

- Uddannelse og vejledning – Vidensbase med en samling af materiale som ledere i kundeservice centeret kan benytte til uddannelse og vejledning af medarbejdere. Materialet leveres af Kundeservicegruppen som en del af det samlede system.

- Market Info – Benyttes til simpel optælling af hvorledes de sager, kundekontaktcenteret håndterer, er fordelt på f.eks. produkter, emner, mv. Når en medarbejder afslutter en sag, skal han/hun angive hvilket produkt, emne, mv. sagen omhandlede.

- Questionaire – Test og analyser der kan benyttes til at vurdere medarbejdernes kompetencer inden for f.eks. bestemte produkter. Denne vurdering vil udføres ud fra, at den enkelte medarbejder fra tid til anden skal gennemgå forskellige tests eller spørgeskemaer.

- IVR – Integration til ekstern service der benyttes til at evaluere medarbejderen efter endt opkald. Dette modul vil ikke have nogen direkte dialog med nogen medarbejdere i kundekontaktcenteret. Modulet vil være kontaktflade til et eksternt system, som kan levere information om kunders tilfredshed.

- Rapportering og Analyse – Generering af rapporter og analyser ud fra data der er opsamlet via de overstående punkter. Disse rapporter vil herefter være grundlag for den løbende kvalitetsmåling. Ud fra rapporterne kan der desuden planlægges f.eks. videreuddannelse af medarbejderne.

1.4 PROBLEMFORMULERING

Efter at have evalueret Tommy Langhoffs beskrivelser af værktøjerne er det vores overbevisning, at man med fordel vil kunne implementere systemet ved hjælp af Service Orientering/Service Orienteret Arkitektur.

Oplægget fra Tommy Langhoff, opstiller som udgangspunkt en række værktøjer/behov for enkeltstående services som hver især knytter sig til en afgrænset arbejdsopgave. Disse hører alle under det samme fagspecifikke domæne, nemlig kundekontaktcenter branchen.

Projektet vil derfor tage udgangspunkt i dette behov og efterprøve, hvorvidt det er fordelagtigt at implementere et system der løser disse behov, ved hjælp af en service orienteret arkitektur. Herunder vil der blive undersøgt, hvilke fordele og ulemper denne type arkitektur medfører.

1.5 SERVICE ORIENTERING / SERVICE ORIENTERET ARKITEKTUR

Der er mange meninger om, hvad begreberne service orientering og service orienteret arkitektur dækker over, samt hvordan disse skal tolkes. Projektet vil ikke fokusere på disse mange holdninger, men forholde sig til begreberne ud fra følgende tilgang: Service orientering er midlet, og service orienteret arkitektur er målet forstået således, at service orientering benyttes til at opstille en service orienteret arkitektur.

Igennem projektet vil der blive arbejdet med en service orienteret tilgang til problemstillingen med henblik på at opstille en service orienteret arkitektur. I projektet vil der blive taget udgangspunkt i, at en service orienteret arkitektur er defineret ud fra de fire hovedregler beskrevet i kapitel 4.

1.5.1 SERVICE ORIENTERING / OBJEKT ORIENTERING

Som ved service orientering og service orienteret arkitektur, er der mange meninger om hvorvidt service orientering og objekt orientering komplementere, overflødiggøre eller direkte er en hindring for hinanden. Projektets indgangsvinkel til denne diskussion er, at service orientering og objekt orientering bør komplementere hinanden for at udvikle systemer med en service orienteret arkitektur.

Service orientering og objekt orientering benyttes på forskellige abstraktions niveauer under analysen af systemet. Service orientering benyttes til at identificere hvilket services systemet skal indeholde og dermed opstiller den service orienterede arkitektur. Når de enkelte services er identificeret vha. service orientering, benyttes der objekt orientering til at udvikle disse services.

1.6 IDENTIFIKATION AF SERVICES I SYSTEMET

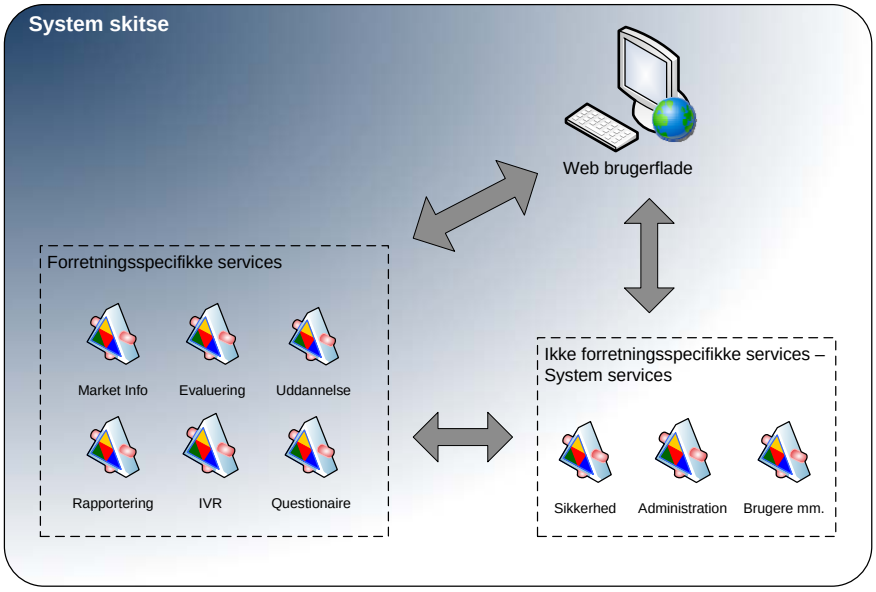
Ud fra de ønskede værktøjer i systemet, kan der identificeres en række services. Hvert af disse værktøjer er helt eller delvist indbyrdes isoleret, det er derfor oplagt, at realisere dem som services der hver især opfylder ét behov.

De opstillede services løser en række forretningsspecifikke behov, men hver af disse services har ligeledes en række praktiske behov.

Et eksempel på dette er brugervalidering. Brugervalidering er ikke direkte et forretningsbehov for kundecenteret, men et mere generelt behov til systemet. Disse vil efterfølgende refereres til som ikke forretningsspecifikke behov.

Disse behov kan ligeledes realiseres som services, disse kaldes for systemservices. Ved at realisere, f.eks. brugervalidering som en systemservice, kan alle de forretningsspecifikke services benytte denne til brugervalidering.

Der er nu opstillet en række services, forretningsspecifikke og ikke forretningsspecifikke, som opfylder hvert deres behov og tilsammen realiserer det ønskede system. Systemet vil kunne skitseres som illustreret i Figur 1.



Figur 1 – System skitse

Ved at sammensætte disse services, er det muligt at opbygge et system der løser kundekontaktcenters behov for et kvalitetssikringssystem. Endnu en fordel ved at benytte en række services er, at systemet nemt kan tilpasses det enkelte kundekontaktcenter, da alle centre ikke nødvendigvis har behov for alle services, eller har ønske til andre og nye services.

Interaktionen med brugeren vil foregå gennem en brugeflade som benytter de pågældende services. Brugerfladen kan, da der benyttes service orienteret arkitektur, teoretisk være implementeret som webside, Windows-applikation, eller en applikation til mobile enheder. Der er dog fra Tommy Langhoffs side ytret ønske om en Web-baseret brugerflade.

1.7 PROJEKT AFGRÆNSNING

Problemstillingen i opgaven er at undersøge, hvorvidt en service orienteret arkitektur med fordel kan benyttes til at opbygge et system til kvalitetssikring i et kundeservicecenter.

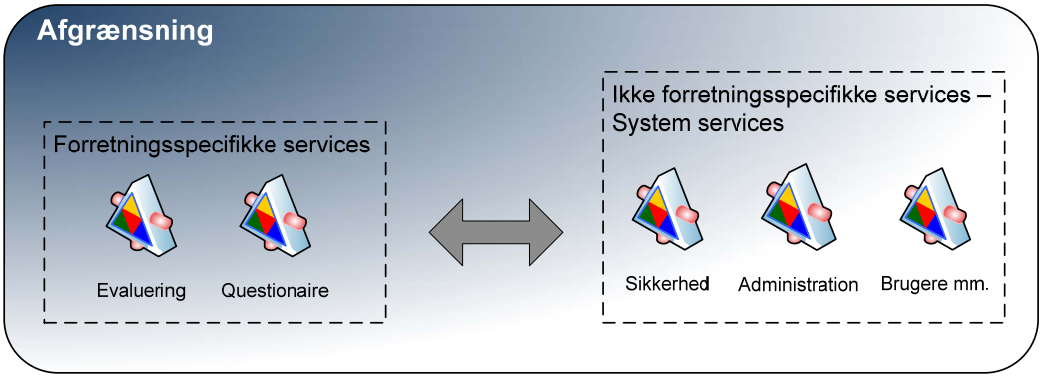
Af hensyn til hvad der i projektperioden er muligt at nå, er projektet begrænset til at fokusere på et udpluk at de nævnte services. Med disse vil vi undersøge hvorvidt service orientering er en velvalgt metode, til at realisere behovene for et system af denne karakter. De udvalgte services er som følger.

- **Evaluering**
- **Questionaire**
- **System**

Kombinationen af disse services vil i projektet blive omtalt som kundekontaktcenter systemet, eller blot systemet.

Fokusområdet for projektet er illustreret i Figur 2

.



Figur 2 – Begrænset systemmodel projekt fokusområde

Der vil ikke blive udviklet brugerflade til systemet, da dette falder uden for projektets fokus område.

Da NextUs er Microsoft Certified Partner og har et godt kendskab til .NET er det på forhånd specificeret, at den service orienterede arkitektur skal implementeres vha. Windows Communication Foundation. Som database system er der valgt Microsoft SQL Server 2005.

Udviklingen vil foregå i Visual Studio 2005 med .NET 2.0 og Windows Communication Foundation Beta 2. Det primære programmeringssprog vil være C#.

1.7.1 TIDSPLAN

I bilag 10 er vedlagt den opstillede tidsplan for projektet.

1.8 RAPPORTENS STRUKTUR

Kapitel 1 – Introduktion

Kapitlet giver en generel introduktion til projektet. Desuden belyses i grove træk den baggrundsviden omkring det fagspecifikke område, som er nødvendig for at forstå grundlaget for udviklingen af systemet.

Kapitel 2 – Kravspecifikation

På baggrund af materiale fra Tommy Langhoff er der udarbejdet en kravspecifikation, som vil blive gennemgået i dette kapitel. Kravspecifikationen vil koncentrere sig om de udvalgte services til projektperioden.

Kapitel 3 – Analyse

Kapitlet vil ud fra de i kapital 2 fastsatte krav beskrive den foretagne analyse. Der vil i den forbindelse blive gennemgået aktører og use case diagrammer. Der vil desuden blive gennemgået en del af de opstillede use Cases i kapitlet. Resten vil af hensyn til læsbarheden være at finde i bilag. I kapitlet vil der også blive gennemgået domæne modeller, som vil ligge til grund for database designet.

Kapitel 4 – Service orienteret arkitektur

I kapitel 4 vil de teoretiske aspekter vedrørende Service orienteret arkitektur blive behandlet. Der vil i kapitlet blive opstillet hvilke fordele, der kan være ved at benytte sig af service orienteret Arkitektur. Kapitlets sidste del vil gennemgå, hvorledes Windows Communication Foundation er opbygget.

Kapitel 5 - Design

Kapitlet gennemgår hvorledes systemet er designet, herunder overordnet arkitektur, sikkerhed og klasse diagrammer.

Kapitel 6 - Implementering

Implementeringsafsnittet vil på baggrund af det fortagende design indeholde de vigtigste aspekter vedrørende implementering. Der vil i kapitlet blive gennemgået de vigtigste punkter vedrørende implementering af systemet ved hjælp af Windows Communication Foundation og .NET.

Kapitel 7 - Test

Der vil i dette kapitel blive opstillet metoder for test af systemet samt eksempler på udførelse af disse.

Kapitel 8 - Konklusion

I konklusionen vil vi opsummere og konkludere på de resultater, vi er kommet frem til set i forhold til de opstillede krav og visioner fra starten af projektet.

Kapitel 2

KRAVSPECIFIKATION

Med udgangspunkt i materialet fra Tommy Langhoff (bilag 1) og begrænsningen i kapitel 1, er der i dette kapitel udarbejdet kravspecifikationen til systemet.

Der vil til hvert område være en kort gennemgang af, hvilken funktion den pågældende del af systemet har. For hver service vil der være en liste af specifikke krav. Kravene er opstillet ud fra materialet, der er modtaget fra Tommy Langhoff. Af hensyn til overblikket i rapporten er den fulde kravspecifikation flyttet til bilag 2.

Der vil i kapitlet være beskrivelse af krav til følgende services:

- **System**
- **Evaluering**
- **Questionaire**

2.1 KRAVSPECIFIKATION FOR SERVICES

2.1.1 SYSTEM

Systemservicen er ikke en service, der direkte fremgår af materialet fra Tommy Langhoff. Denne service er indført for at opfylde en række ikke funktionelle krav, som beskrevet i afsnit 1.6. Denne service skal samle de administrative funktioner og stille dem til rådighed for andre services i systemet.

Den primære opgave vil være at håndtere bruger administration. Servicen skal tilbyde operationer, der gør det muligt at oprette, slette og vedligeholde brugere, grupper og roller. Desuden skal servicen tilbyde brugervalidering og autorisering samt styring af hvilke services, der udgør systemet.

Systemservicens behov er delt op i følgende grupper, som beskrives i det følgende. Der er indsat figurer, som for hver gruppe viser de opstillede krav. De specifikke krav for de enkelte grupper er at finde på tabelform i bilag 2

1. Adgangskontrol
2. Bruger administration
3. Gruppe administration
4. Rolle administration
5. Service administration

2.1.1.1 Adgangskontrol.

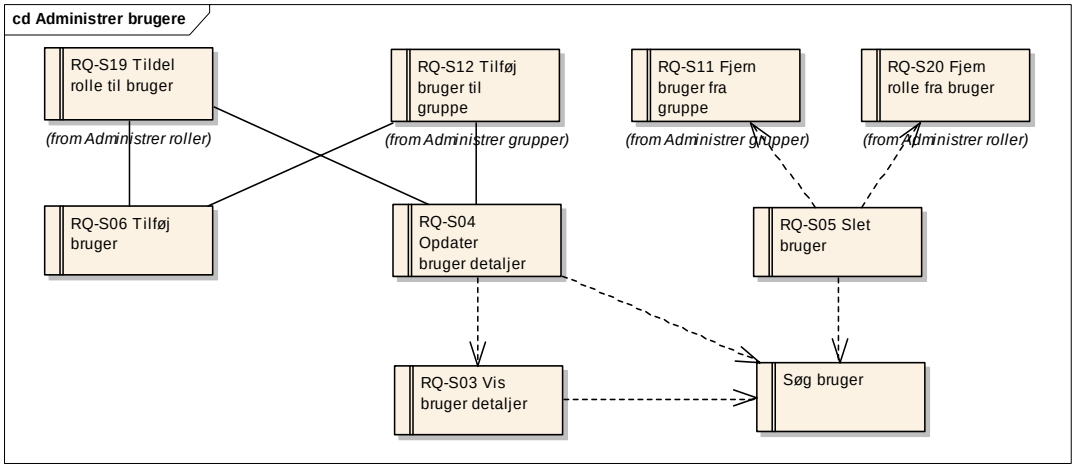
Adgangskontrol skal implementere funktionalitet, der gør det muligt for andre services at validere en bruger imod systemet, samt autorisere om den pågældende bruger må anvende den operation der forsøges benyttet.

Tabel 1 – Krav til adgangskontrol i systemservicen

Navn	Beskrivelse
RQ-S01 Valider bruger	Det skal være muligt at validere en bruger i systemet på baggrund af brugernavn og password.
RQ-S02 Autoriser bruger	Det skal være muligt at checke hvor vidt en bruger har tilladelse til at udføre en bestemt operation ud fra de roller brugeren er tildelt.

2.1.1.2 Administrer brugere.

Gruppen Administrer bruger indeholder de krav, der er stillet til systemet, set i forhold til at kunne håndtere brugere.

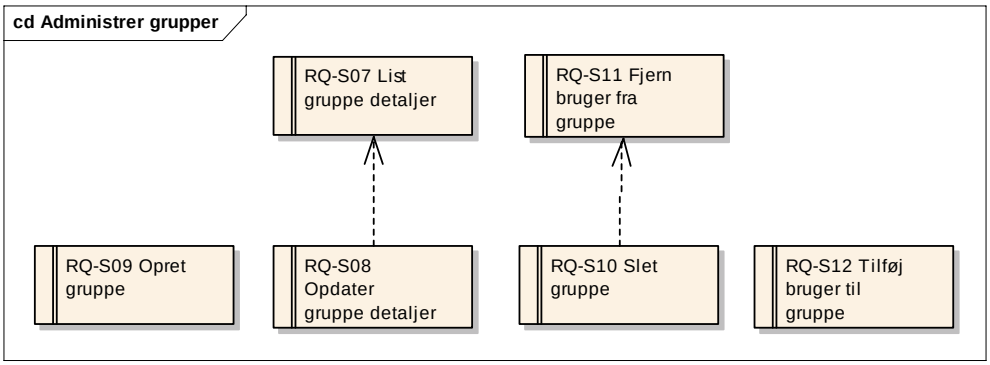


Figur 3 – Kravsmodel for bruger administration

I Figur 3 er vist de krav der er opstillet til bruger administration i systemet. Detaljer kan findes i bilag 9.2.1.2

2.1.1.3 Administrer grupper.

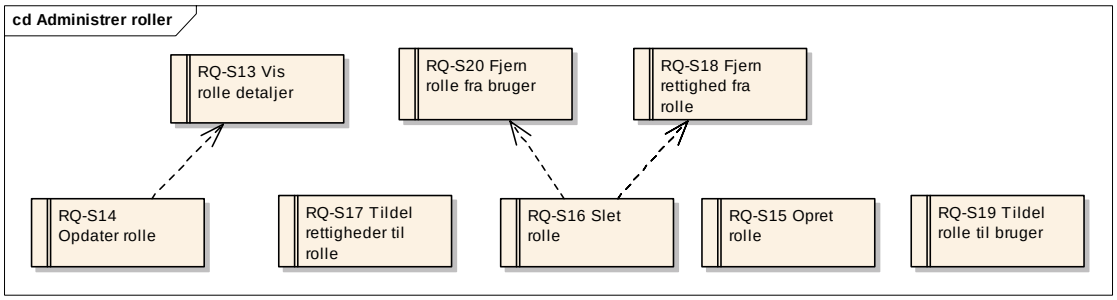
For at imødekomme behovet for inddeling af brugere i forskellige grupper er der opstillet de i Figur 4 viste krav til systemet. Detaljer kan findes i bilag 2 afsnit 9.2.1.3



Figur 4 – Kravsmodel for gruppe administration

2.1.1.4 Administrer roller

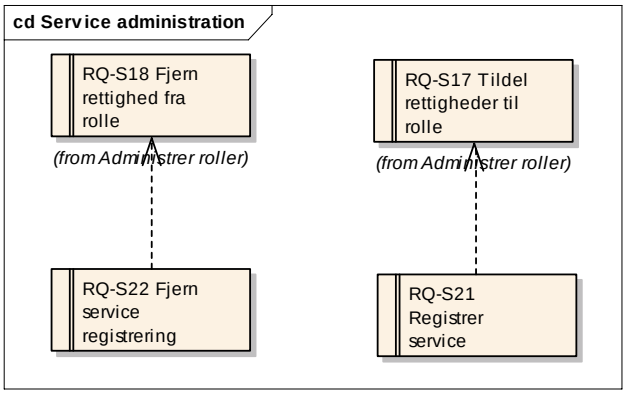
For at Systemservicen har mulighed for at administrere hvilke brugere, der kan udføre de forskellige operationer i systemet, skal det ligeledes understøtte roller. Vi har valgt at arbejde med rolle strukturen, da vi er bekendt med den, og fordi den er godt understøttet i Microsoft platformen, som vi har valgt at bygge vores løsning på. Systemservicen skal derfor understøtte, at man kan operere med roller, som tildeles rettigheder. Brugere kan herefter tildeles roller. De nødvendige krav er vist i Figur 5 herunder. Detaljer kan findes i bilag 2 afsnit 9.2.1.4



Figur 5 – Kravsmodel for rolle administration

2.1.1.5 Service-administration.

Når der implementeres ny funktionalitet i systemet i form af nye services, skal disse kunne tilføjes systemet. Systemservicen skal derfor understøtte tilføjelse af en ny service og importere information om den nye services operationer, således at brugere kan tildeles roller og derigennem opnå rettighed til at bruge den nye service og dens operationer. De nødvendige krav til denne del af systemet er vist i Figur 6. Detaljer kan findes i bilag 2 afsnit 9.2.1.5



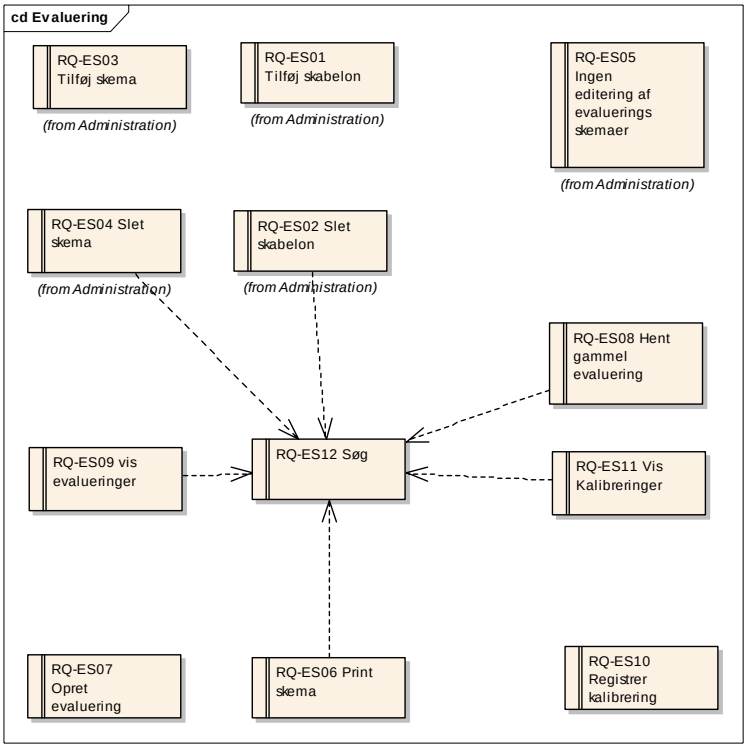
Figur 6 – Kravsmodel for service administration

2.1.2 EVALUERING

Denne service anvendes i forbindelse med at en medarbejder skal evalueres. Servicen skal realisere behovet beskrevet i bilag 1 afsnit 9.1.3. Servicen skal i grove træk gøre det muligt at oprette en række skemaer af forskellig karakter. Disse skemaer bruges som retningslinjer og dokumentation for evaluering af medarbejdere. Som processen er beskrevet i bilag 1 afsnit 9.1.3. vedrørende processen *Samlyt*, skal det være muligt at printe skemaer, så de kan anvendes uden brug af computer. Det skal desuden være muligt at indtaste en evaluering i skemaerne og gemme den i systemet.

Ud over at tilbyde materiale til evaluering af medarbejdere, skal servicen ligeledes tilbyde en registrerings-funktion til "lederkalibrering". Lederkalibrering er et koncept, hvor lederne mødes og sammen evaluerer en række medarbejdere. Formålet er at ensrette ledernes opfattelse af de enkelte medarbejdere, således at der foretages en ensartet evaluering af medarbejdere på tværs af lederne. Servicen skal tilbyde mulighed for at registrere disse kalibreringer. Begrebet kalibrering forklares yderligere i analysen i næste kapitel.

Til servicen Evaluering er der defineret en række krav vist i Figur 7. Figuren viser den grafiske opstilling af kravene samt deres relationer. Detaljer kan findes i bilag 2 afsnit 9.2.2

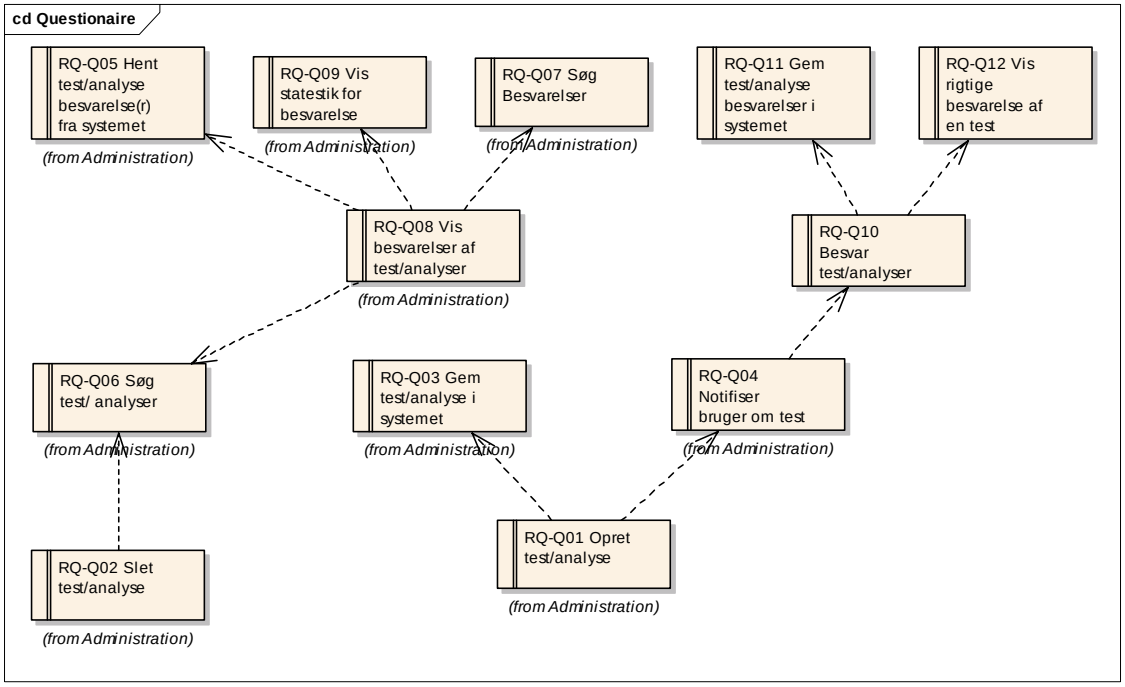


Figur 7 – Kravsmodel for Evalueringservicen

Der er i hver service knyttet nogle krav til en administrationsdel. Disse krav er blevet markeret med *(from Administration)***.** Disse krav tilhører en kategori af krav, der henvender sig mere til administration end til anvendelse af servicen. Disse krav vil være forbeholdt en gruppe af brugere, der er tildelt et højere niveau af rettigheder end den almene bruger. Resten af kravene i Figur 7 er knyttet til den almindelige brug af servicen.

2.1.3 QUESTIONAIRE

Questionaireservicens primære opgave er, at gøre det muligt for lederne at udrulle tests og analyser til sine medarbejdere. Det skal derfor ved hjælp af denne service være muligt at oprette tests og analyser med spørgsmål og svar. Lederen skal efter oprettelse kunne tildele dem til de ønskede brugere og grupper. Lederen skal efterfølgende kunne følge udviklingen af besvarelser samt se besvarelserne. Kravene til servicen er vist i Figur 8. På samme måde som Evalueringservicen er der visse krav, der er markeret med Administration, da de skal realisere administrations relaterede funktioner i servicen. Detaljer kan findes i bilag 2 afsnit 9.2.3



Figur 8 – Kravsmodel for Questionaireservicen

2.2 RESUME

Der er her i kravspecifikationen løseligt introduceret en mængde af begreber der knytter sig til det system der skal udvikles.

2.2.1 SYSTEMSERVICE

Krav til systemservicen er, at den skal understøtte administration af brugere. Flere brugere kan administreres samtidig ved hjælp af roller. I systemservicen skal man kunne "indmelde" services i systemet og få oprettet lister med de operationer, systemets services tilbyder. Ud fra disse lister med operationer kan der oprettes roller, der tildeles rettigheder i form af operationer fra systemets services. Brugere kan tildeles forskellige roller og derigennem få adgang til systemets forskellige funktionaliteter.

2.2.2 EVALUERING

Evaluerings-servicen indfører begreber som skemaer, protokoller og kalibreringer. Servicen skal tilbyde mulighed for at oprette protokoller, som skal være guidelines for ledere, når de evaluerer medarbejdere. Når medarbejderne er evalueret udfyldes Feedback skemaer, som ligeledes skal kunne administreres i servicen. For at lederne på tværs af en organisation kan have en ensartet opfattelse af hvad god kvalitet er, afholdes kalibreringer hvor lederne sammen evaluerer nogle medarbejdere for at opnå en ensartet opfattelse. Disse kalibreringer skal ligeledes kunne registreres i systemet.

2.2.3 QUESTINARIE

For at lederne i kundekontakt-centre kan holde sig ajour med hvilket niveau, der generelt er i deres afdelinger, skal lederne kunne oprette analyser og tests, som kan udrulles til medarbejderne. En test er en samling spørgsmål medarbejderen skal svare på. Til testene er desuden defineret rigtige svar. Analysen er identisk med testen bortset fra, at der ikke er defineret rigtige svar på spørgsmålene.

Ud fra materialet fra Tommy Langhoff er der udarbejdet kravspecifikation for servicerne i systemet. Denne kravspecifikation vil ligge til grund for den videre udvikling af systemet.

Kapitel 3

ANALYSE

I kapitlet er der på baggrund af kravspecifikationen fra kapitel 2 og materialet fra Tommy Langhoff udført analyse, hvor der er opstillet aktør definitioner, use cases og domæne modeller.

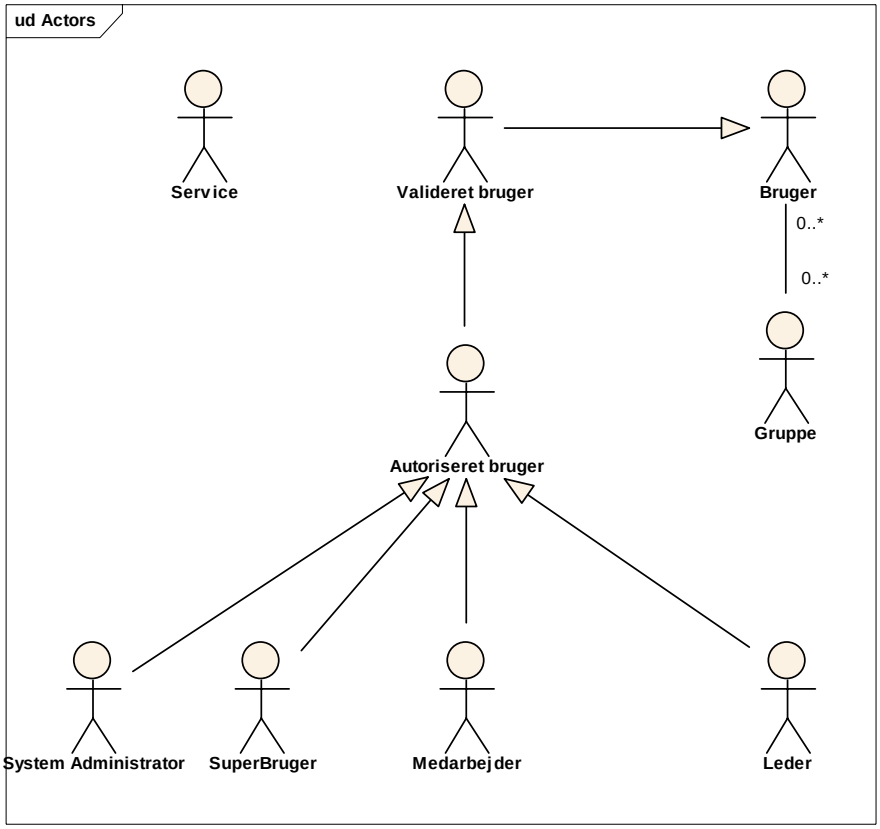
For hver service vil domæne modellerne blive gennemgået, og der vil være en overordnet gennemgang af use case modellerne. For Questionaireservicen vil der ligeledes være en komplet gennemgang af use cases. Use cases for system og evalueringservicerne vil være at finde i bilag.

Analysen skal desuden klarlægge rigtigheden af antagelserne i afsnit 1.6 om, hvorvidt behovene i et kundeservicekontaktcenter kan realiseres som individuelle services, er korrekte.

3.1 AKTØRER

Til systemet er der opstillet en samling aktører. Aktører er definitioner på personer, services eller andre elementer, der interagerer med systemet. Aktørerne anvendes når der opstilles use cases for at definere hvem eller hvad, der interagerer med systemet i en given situation.[2] I systemet vil brugere operere på flere niveauer med forskellige rettigheder. I disse situationer er det også nyttigt at have en sådan hierarkisk inddeling af aktører. Der er på baggrund af disse overvejelser opstillet aktører. Aktørerne knytter sig hierarkisk sammen som vist i Figur 9, vist herunder.

3.1.1 AKTØR DIAGRAM



Figur 9 – Aktør diagram

3.1.2 AKTØR BESKRIVELSER

Bruger
Person der er oprettet som bruger i systemet.

Valideret bruger
En bruger der er succesfuldt valideret af systemet ud fra brugernavn og password.

Autoriseret bruger
En valideret bruger der er blevet autoriseret af systemet til at udføre en operation.

Gruppe
Samling af brugere til organisatoriske formål. En gruppe kan have en leder tilknyttet, som er ansvarlig for gruppen. Gruppen består af et vilkårligt antal af systemets aktører.

Leder
Lederen står for den daglige styring af medarbejderne og foretager evalueringer og test/analyser af medarbejderne. En leder er typisk leder for en gruppe af flere medarbejdere.

Medarbejder
En medarbejder er en valideret og autoriseret bruger, som til dagligt benytter systemet i forbindelse med deres arbejde med kundekontakt. Ofte omtalt som "Caller"

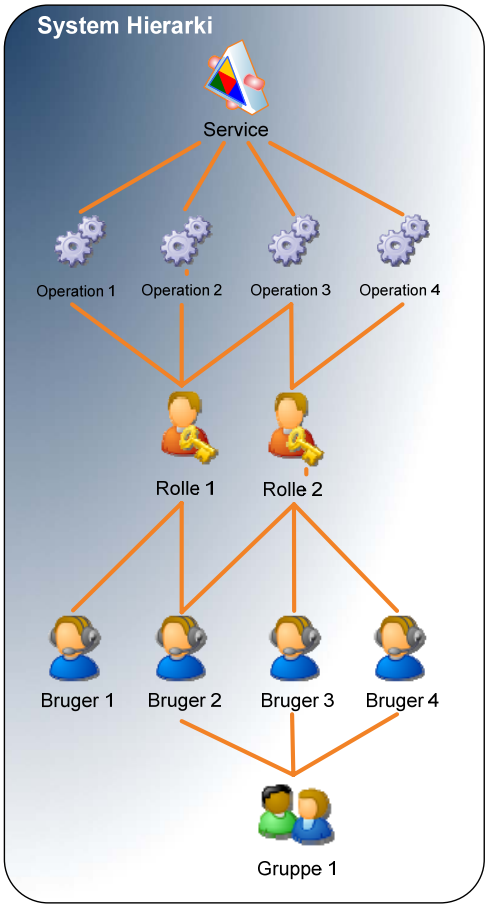
Service
En service afspejler de services, som systemet er bygget op af.

Superbruger
En superbruger er en bruger, typisk en leder, som har fået tildelt yderligere administrative rettigheder til systemet. Som f.eks. at oprette brugere, grupper mv.

System Administrator
System Administratoren har fulde administrative rettigheder til systemet. Det vil typisk være System Administratoren der står for driften af systemet.

3.2 SYSTEM SERVICE

Systemservicen skal realisere de ikke funktionelle krav såsom administration af brugere, grupper, roller, rettigheder og services.



Figur 10 – Hierarki i systemservicen

Hvorledes disse elementer relaterer til hinanden kan ses i Figur 10. Figuren illustrerer, hvorledes en række brugere kan være medlem af en gruppe. Dette kan f.eks. være på baggrund af en bestemt produkttype, kundekontaktcenteret supporterer.

Brugere kan ligeledes tildeles roller, som giver dem rettigheder til at benytte udvalgte operationer i services.

Systemet skal ud over at stå for administrationen af dette, ligeledes håndtere brugervalidering og autorisation ud fra roller, brugere er tildelt.

Der er på baggrund af ovenstående funktionalitet blevet lavet følgende punkter, som opdeler analysen for systemservicen.

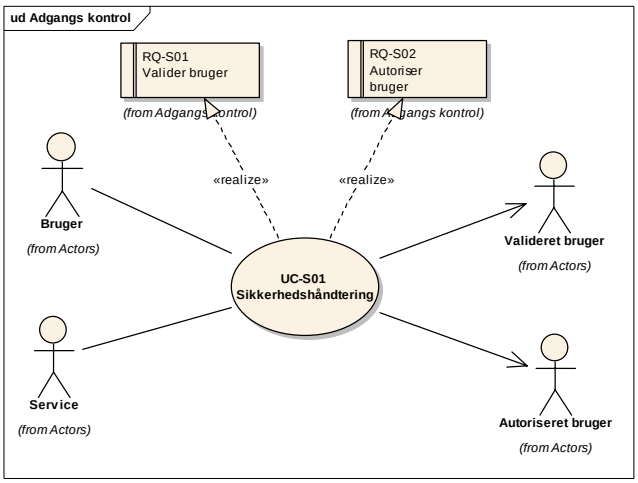
- Adgangs-kontrol.
- Bruger-administration
- Gruppe-administration
- Rolle-administration
- Service-administration

3.2.1.1 Adgangskontrol

I adgangskontrol er der defineret en use case *UC-S01 Sikkerhedshåndtering*. Denne use case, hvis relationer til aktørerne er vist i use case diagrammet i Figur 11, dækker validering og autorisering af brugere.

Use cases i modellen:

- UC-S01 Sikkerhedshåndtering - bilag 3 afsnit 9.3.3.1.1



Figur 11 – Use case diagram til sikkerhedshåndtering

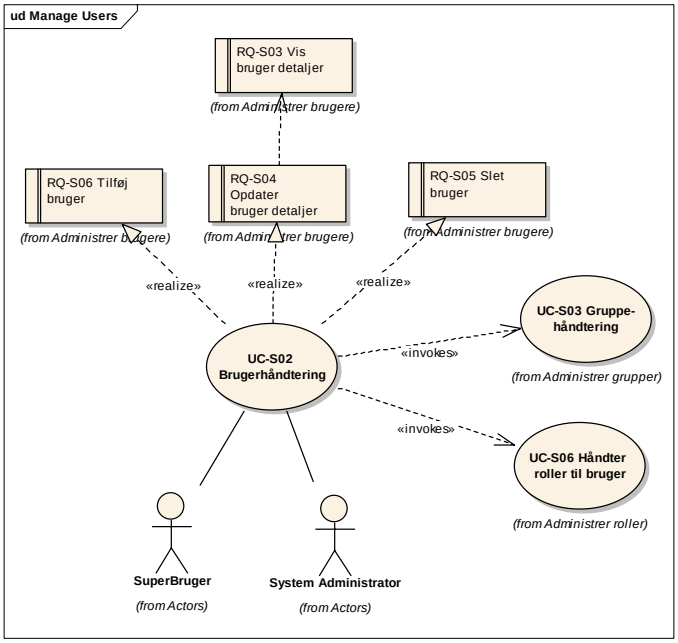
Sikkerhedshåndterings use casen benyttes af brugere til at ændre status fra ukendt til valideret og autoriseret, således brugeren kan anvende systemet. En aktør ændrer altså status ved kontakt med denne use case.

3.2.1.2 Brugeradministration

I bruger administrationsdelen er der opstillet *UC-S02 Brugerhåndtering*. I denne dækkes scenarier angående oprettelse af nye brugere, sletning af brugere samt opdatering af brugere.

Use cases i modellen:

- UC-S02 Brugerhåndtering - bilag 3 afsnit 9.3.3.2



Figur 12 – Use case diagram brugeradministration

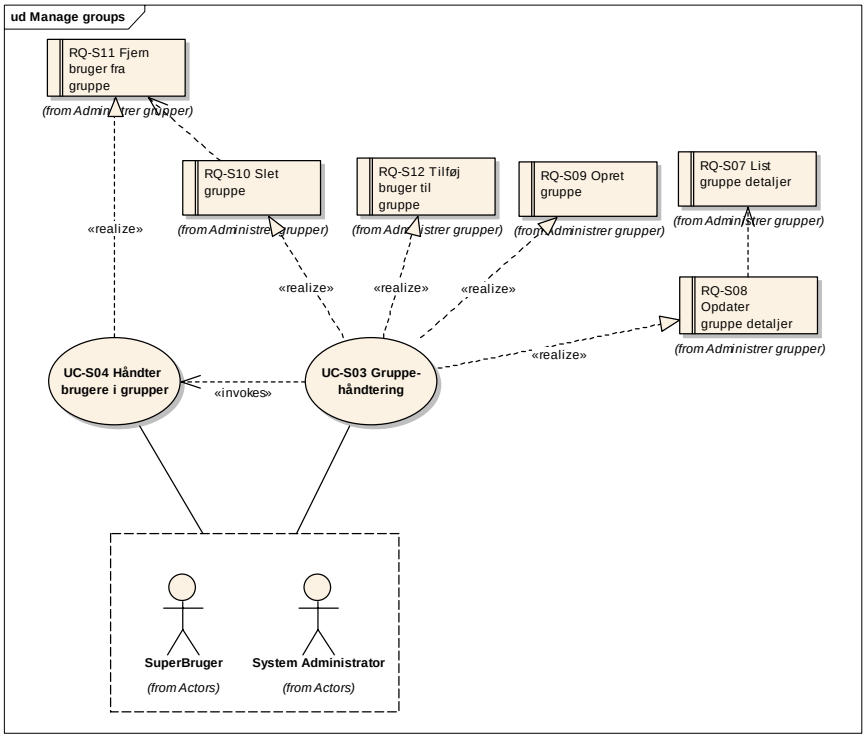
Som det ses i Figur 12 findes der relationer til *UC-S03* og *UC-S06*. Dette skyldes, at når en bruger slettes skal denne desuden fjernes fra grupper, hvilket *UC-S03* er ansvarlig for. Det samme gør sig gældende med brugerens relationer til roller, som håndteres af *UC-S06*.

3.2.1.3 Gruppeadministration

Til gruppe administration er der defineret 2 use cases, som henholdsvis dækker scenarier vedrørende at slette, oprette og opdatere grupper og scenarier, der omhandler administration af brugers tilhørsforhold til grupper.

Use cases i modellen

- UC-S03 Gruppe håndtering - bilag 3 afsnit 9.3.3.3.1
- UC-S04 Håndter brugere i grupper - bilag 3 afsnit 9.3.3.3.3



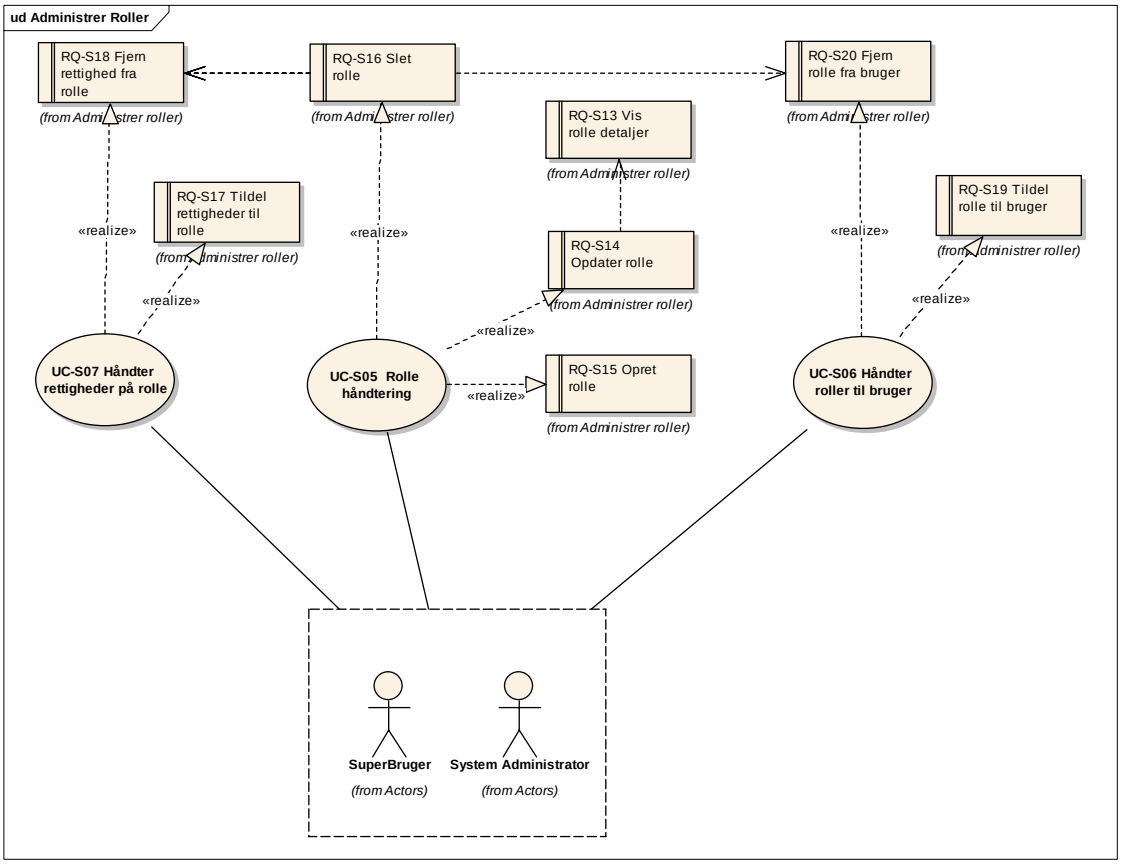
Figur 13 – Use case diagram for gruppeadministration

3.2.1.4 Rolle-administration

Der er i systemet valgt at basere rettighedsstyringen på roller. Det vil sige, at der ikke tildeles rettigheder til brugere direkte. Der oprettes roller i systemet, rollerne tildeles herefter forskellige rettigheder, og brugere og grupper tildeles derefter roller. Der er til at realisere disse krav opstillet følgende Use cases.

Use cases i modellen:

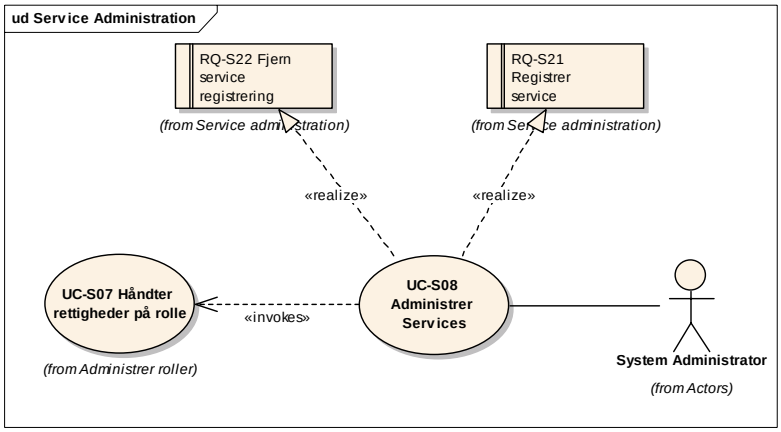
- UC-S05 Rolle håndtering - bilag 3 afsnit 9.3.3.4.1
- UC-S06 Håndter roller til brugere - bilag 3 afsnit 9.3.3.4.2
- UC-S07 Håndter rettigheder og roller - bilag 3 afsnit 9.3.3.4.3



Figur 14 – Use case diagram for rolle administration

3.2.1.5 Service Administration

For at kunne styre hvilke rettigheder brugerne skal tildeles er det nødvendigt, at systemservicen ved hvilke operationer, systemet tilbyder. På grund af dette, er der oprette use cases, hvor det er defineret, hvorledes en service tilføjes og fjernes fra systemet. UC-S08 Administrer services - bilag 3 afsnit 9.3.3.5.1

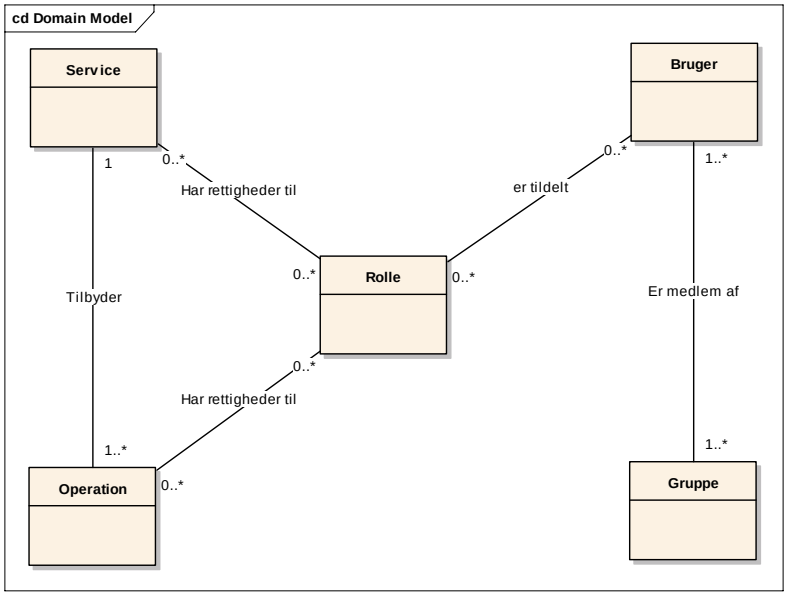


Figur 15 – Use case diagram service administration

Som det ses i Figur 15 er der relationer til *UC-S07*. Dette skyldes, at der i forbindelse med tilføjelse eller fjernelse af services fra systemet også vil blive lavet om på rettighederne i rollerne, hvilket bliver håndteret i *UC-S07*.

3.2.2 DOMÆNEMODEL

Der er til systemservicen lavet domænemodel, der klarlægger begreberne, der knytter sig til systemservicen. Disse begreber er identiske med, dem der blev indført i Figur 10, da denne er lavet på baggrund af domænemodellen.



Figur 16 – Domænemodel til systemservice

3.3 EVALUERING

Servicen evaluering er et værktøj til praktisk styring af gennemførelse af *samlyt*, lagring af data og opfølgning på samlyt.

Samlyt er en betegnelse for en proces, hvor en leder overværer, at en medarbejder er i dialog med kunden. På baggrund af et antal af disse iagttagelser pointgiver lederen medarbejderen ud fra en i forvejen udformet protokol.

Protokollen er et skema, hvor man har opdelt samtaler i forskellige faser. Dette kan være kontaktfase, behovsdækning, løsning af problemer og lignende. Under hver fase er der punkter som medarbejderen bliver evalueret ud fra. Hvert punkt har en scoring og en vægt. Vægtningen er et udtryk hvor stor betydning det enkelte punkt har samlet i evalueringen. Et brudstykke af en protokol kan se ud som herunder.

Tabel 2 – Forskal til protokol i Evalueringservicen

Fase	Scoring	Vægt	Samlet
Kontakt fasen • Velkomst • Præsentation Samlet	 5 4 4,5	 30%	 1,35
Behovsafdækning • Spørgeteknik • Behov afdækket ok Samlet	 3 3 3	 70%	 2,10
Løsning			

Når denne del af processen er foretaget og protokollen er udfyldt, skal lederen give medarbejderen feedback. Dette gøres ved, at lederen igen anvender et i forvejen defineret skema, denne gang et feedbackskema.

Feedbackskemaet er et skema, der skal indeholde lederens konklusioner og anbefalinger ud fra, hvad lederen oplevede under samtalerne. Det vil også indeholde information om, hvad der kan være afledt af evalueringen, f.eks. uddannelse eller lignende.

For at lederne i en organisation har en ensartet evaluering af medarbejderne er der et behov for, at de indbyrdes afstemmer deres vurderinger af hvad god kundekontakt indebærer. Til løsning af dette har Tommy Langhoff indført lederkalibrering. Lederkalibrering dækker over en rutine, hvor lederne sætter sig sammen. Hver leder pointgiver de samme optagede samtaler mellem medarbejder og kunde. Herefter diskuteres disse for at opnå enighed omkring en ensartet pointgivning af den enkelte præstation. Sådanne kalibreringer skal kunne registreres i servicen.

For at sammenholde information omkring samlyt-processen, er der indført begrebet evaluering. En Evaluering skal indeholde alt, hvad der hører til processen. Herunder generel information om hvilken medarbejder der er evalueret, hvor mange samtaler der er evalueret, hvornår evalueringen er foretaget, hvem der har fortaget den og så fremdeles. Ud over dette, skal der også være arkiveret udfyldte skemaer, der har været anvendt i processen, herunder protokoller og feedback-skemaer.

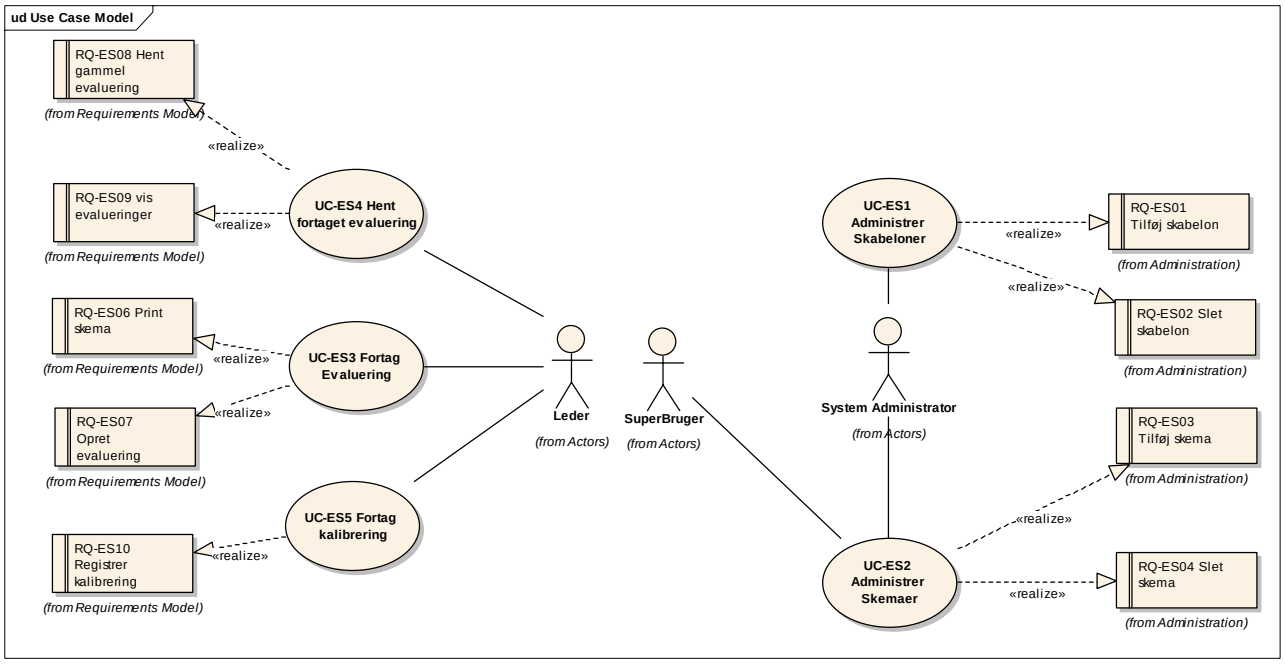
3.3.1 USE CASE MODEL

Ud fra de behov der er til servicen, er der oprettet use cases, som vist i Figur 17 herunder

Der er for Evaluerings-servicen opstillet følgende use cases:

- UC-ES1 Administrer Skabeloner – bilag 3 afsnit 9.3.1.1
- UC-ES2 Administrer Skemaer – bilag 3 afsnit 9.3.1.2
- UC-ES3 Foretag evaluering – bilag 3 afsnit 9.3.1.3
- UC-ES4 Hent foretaget evaluering – bilag 3 afsnit 9.3.1.4
- UC-ES5 Foretag kalibrering – bilag 3 afsnit 9.3.1.5

3.3.1.1 Use case diagram:

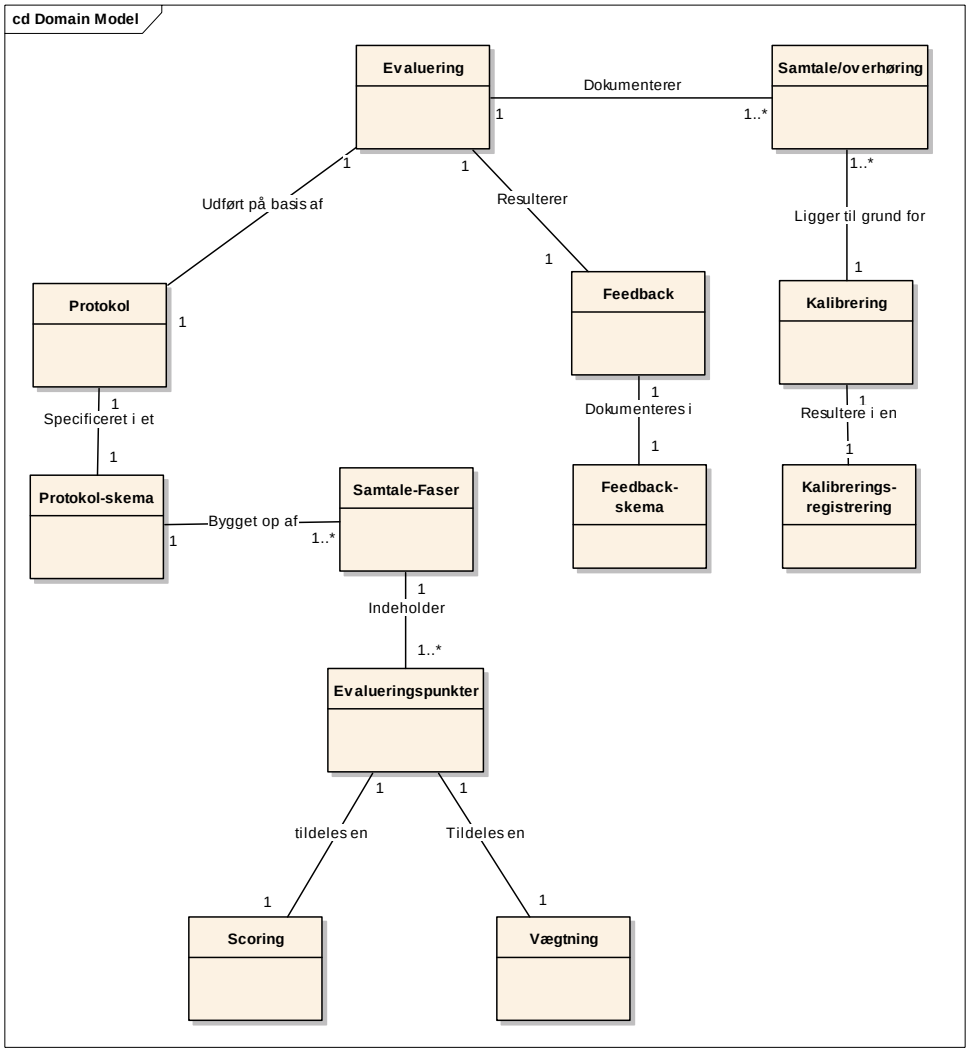


Figur 17 – Use case diagram Evalueringservice

I højre side af Figur 17 er der defineret use cases til administration af skemaer og skabeloner. I venstre side er de use cases der henvender sig til den generelle anvendelse af servicen - at foretage evalueringer af medarbejdere og registrere disse i systemet, samt at registrere kalibreringer.

3.3.2 DOMÆNEMODEL

Ud fra arbejdet med at definere formålet med servicen, er der blevet defineret et antal begreber. For at få et overblik over sammenhængen mellem disse begreber er der udformet følgende domænemodel. Denne model vil også være grundlag for modellering af database til servicen. Det er i figuren vist, hvordan de forskellige begreber hænger sammen.



Figur 18 – Domænemodel til evalueringsservice

3.4 QUESTIONAIRE

Kundekontaktcentre har et behov, for konstant at kunne vurdere deres medarbejderes kompetenceniveauer inden for f.eks. et specifikt produktområde, således at de altid kan yde den bedste service for deres kunder.

Questionaireservicen gør det muligt for ledere i kundekontaktcentre at stille medarbejderne en række spørgsmål udformet som enten en test eller en analyse. Analyser vil typisk være interne analyser, som f.eks. medarbejder tilfredshedsanalyser. Tests vil kunne benyttes til at vurdere medarbejdernes kompetenceniveauer.

Begge typer er bygget op som et multiple choice spørgeskema, med et antal spørgsmål og et antal svar muligheder til hvert spørgsmål. Testen er forskellig fra analysen, ved at der på forhånd er angivet rigtige svar.

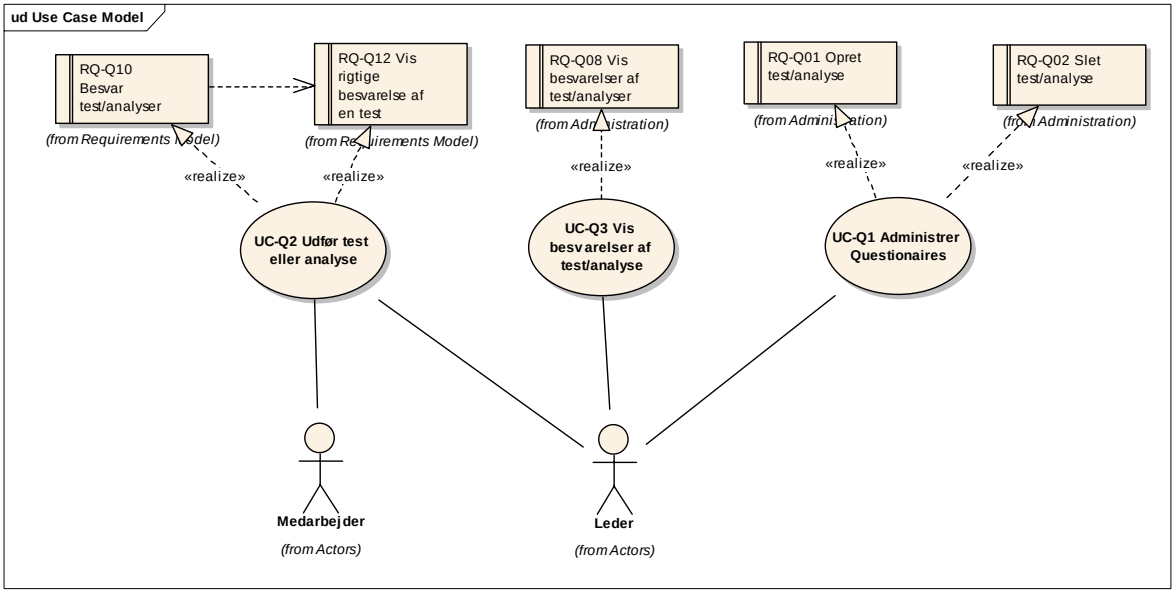
En test eller analyse kan tildeles specifikt til en bestemt målgruppe. Dette kunne f.eks. være en gruppe medarbejdere inden for et bestemt produktområde, eller i en bestemt afdeling. Desuden skal det være muligt at angive, hvorvidt en test/analyse kan besvares anonymt og ligeledes en tidsfrist for, hvornår den senest skal besvares.

3.4.1 USE CASE MODEL

De oprettede use cases til dette modul er følgende:

- UC-Q1 Administrer Questionaires – bilag 3 afsnit 9.3.2.1
- UC-Q2 Vis besvarelse af test/analyse – bilag 3 afsnit 9.3.2.2
- UC-Q3 Udfør test eller analyse – bilag 3 afsnit 9.3.2.3

3.4.1.1 Use case diagram:



Figur 19 – Use case diagram for Questinaire servicen

I venstre side af Figur 19 er *UC-Q2* Udfør test/analyse. Denne henvender sig til medarbejderne og dækker over den generelle anvendelse af servicen. I højre side af figuren er de to use cases *UC-Q1* og *UC-Q3,* som henvender sig til lederne og dækker over de administrative funktioner i servicen med henblik på oprettelse og udrulning af tests og analyser. De 3 use cases er indsat herefter.

3.4.1.2 UC-Q1 Administrer Questionaires

Navn:	**UC-Q1 Administrer Questionaires**	
Forfatter	**Peter Thornton**	
Noter:	**Mål:** **At foretage administrative opgaver af tests og analyser.** **Primær aktør:** **Leder**	
Scenarier:		
Type:	**Formål:**	**Beskrivelse:**
Succes	Oprettelse af analyse	Aktøren ønsker at oprette en analyse og skal angive følgende 1: Spørgsmål og tilhørende svarmuligheder. 2: Tidsfrist i klokkeslæt og dato for seneste besvarelse af analysen. 3: Hvilke brugere og/eller grupper der skal gennemføre testen. 4: Om besvarelser skal være anonyme eller ej. Når aktøren har angivet ovenstående, gemmes analysen i systemet og den udgives, således at analysen er tilgængelig for de i punkt 3 angivne brugere, således de kan besvare analysen.
Succes	Slet test/analyse	En aktør vælger en eksisterende test/analyse der ønskes slettet. Aktøren promptes for en bekræftelse af at denne test/analyse ønskes slettet. Hvis dette bekræftes slettes testen/analysen fra systemet
Alternativ	Udvidelser	Oprettelse af analyse: Hvis aktøren under oprettelse af en analyse fortryder oprettelsen, skal systemet returnere til tilstanden før oprettelsen blev påbegyndt. Evt. data der er gemt i systemet skal slettes.
Constraints:		
Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal have autorisation.	Brugeren skal være autoriseret til at oprette en analyse
Post-condition	Den ønskede operation er udført	

3.4.1.3 *UC-Q2 Udfør test eller analyse*

Navn:	**UC-Q2 Udfør test eller analyse**
Forfatter	**Peter Thornton**
Noter:	**Mål:** **At gennemføre en test eller analyse** **Primær aktør:** **Medarbejder**

Scenarier:

Type:	**Formål:**	**Beskrivelse:**
Succes	Gennemfør analyse/test	1: Aktøren ønsker at besvare en analyse. 2: Der svares på et spørgsmål ad gangen, indtil alle spørgsmål er besvaret. 3: Når alle spørgsmål er besvaret, skal aktøren bekræfte at besvarelsen gemmes i systemet. 4: Besvarelsen gemmes i systemet med information om hvornår besvarelsen er foretaget. 5: Hvis en aktør har afsluttet en test, skal de rigtige svar til spørgsmålene præsenteres. Følgende præsenteres for brugeren: - Hvor mange rigtige svar aktøren havde ud af hvor mange, testen indeholdte - Hvilket spørgsmål aktøren havde svaret forkert på - De rigtige svar til de spørgsmål, aktøren havde svaret forkert på
Alternativ	Udvidelser	2a: Aktøren kan gå tilbage til de tidligere spørgsmål og ændre sin besvarelse. 2b: Aktøren kan afbryde besvarelsen, og vil blive præsenteret med muligheden for enten at: 1) Gemme besvarelsen hvor han/hun er nået til, og senere genoptage besvarelsen fra dette punkt. 2) Ignorere allerede besvarede spørgsmål og starte forfra på besvarelsen senere. 3a: Hvis analysen ikke er angivet som anonym, informeres aktøren om, at det vil fremgå af besvarelsen, hvem der har udført den. 4a: Hvis analysen ikke er angivet som anonym gemmes ligeledes hvilke aktører, der har udført besvarelsen.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren skal valideres af systemet
Pre-condition	Brugeren er autoriseret	Brugeren skal autoriseres i systemet til at besvare den givne analyse
Pre-condition	Brugeren er i målgruppen for analysen	Den givne analyse er tildelt til besvarelse af brugeren, eller en gruppe brugeren er medlem af.
Pre-condition	Analysen/testen er gennemført	Den udgivne analyse/test er besvaret og besvarelsen er gemt i systemet

3.4.1.4 *UC-Q3 Vis besvarelser af test/analyse*

Navn:	**UC-Q3 Vis besvarelser af test/analyse**
Forfatter	**Peter Thornton**
Noter:	**Mål:** **At vise besvarelserne til en test/analyse** **Primær aktør:** **Leder**

Scenarier:

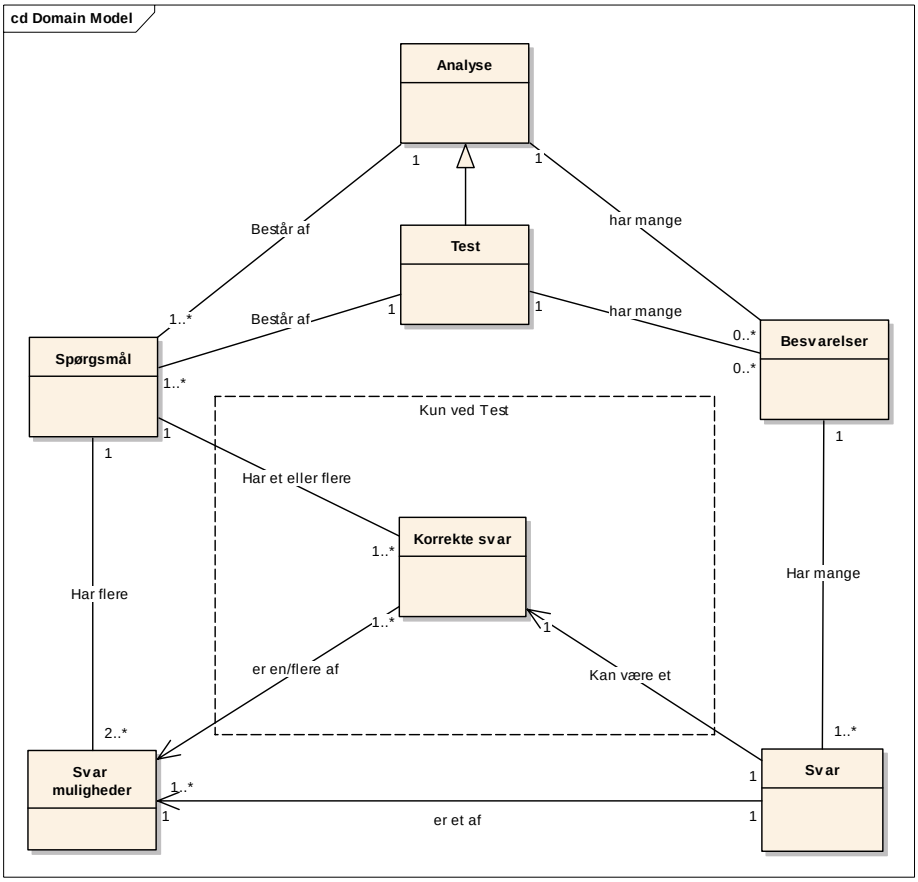
Type:	**Formål:**	**Beskrivelse:**
Alternativ	Anonyme besvarelser	Hvis testen/analysen er angivet som anonym, skal det ikke fremgå af præsentationen af besvarelserne, hvordan de enkelte brugere har svaret. Er testen/analysen ikke angivet som anonym, skal det fremgå, hvordan de enkelte brugere har svaret.
Succes	Vis besvarelser	En aktør vælger en eksisterende test/analyse og ønsker at se besvarelserne til denne. Besvarelserne præsenteres herefter for aktøren.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren skal være valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal være autoriseret	Brugeren skal være autoriseret til at se besvarelser af en test/analyse
Post-condition	Besvarelser er vist	De gemte besvarelser bliver vist på skærmen med en passende præsentation.

3.4.2 DOMÆNEMODEL

Ud fra materialet omkring processen i test og analyse servicen er der opstillet en domænemodel som vist i Figur 20. Figuren viser hvordan de forskellige begreber, der knytter sig til servicen, hænger sammen.



Figur 20 – Domænemodel for Questionaireservicen

Af domænemodellen fremgår det, at en test og en analyse er meget lig hinanden, og kun adskiller sig ved de rigtige svar til testen. Af samme årsag er begge dele samlet i en domæne model, hvor forskellen er illustreret med en stiplet boks.

3.5 RESUME

Hver service er analyseret ud fra objekt orienterede principper beskrevet i [2] og [3]. Der er opstillet domæne modeller og use cases for disse services.

Som beskrevet i indledningen til kapitlet, er hver service analyseret separat som et isoleret domæne. Der er i gennem analysen ikke fremkommet nogle problematikker eller overlap mellem servicerne, andet end de allerede definerede fællestræk, som er samlet i systemservicen. Dette underbygger derfor beslutningen om at behandle hver service separat. Denne tydelige veldefinerede grænse i hver service har gjort det muligt, at analysere hver service ud fra objekt orienterede principper og samtidig underbygget muligheden for at gennemføre Service Orientering af hele løsningen.

Der er ligeledes ikke opstået problemer ved opstillingen af aktør definitioner. Dette underbygger beslutningen om, at opstille disse for hele systemet, således de dækker alle services i stedet for at opstille dem separat for hver service.

Kapitel 4

SERVICE ORIENTERET ARKITEKTUR

I dette kapitel vil det blive gennemgået, hvad begrebet service orienteret arkitektur dækker over. Der vil være en generel gennemgang efterfulgt af de 4 hovedregler i service orienteret arkitektur. Første del af kapitlet afrundes med en opsummering af karakteristika ved service orienteret arkitektur.

Anden del af kapitlet fokuserer på Windows Communication Foundation (WCF). Der gennemgås her hvorledes WCF er bygget op, samt hvorledes det i sin natur understøtter og hjælper til med implementering af en service orienteret arkitektur.

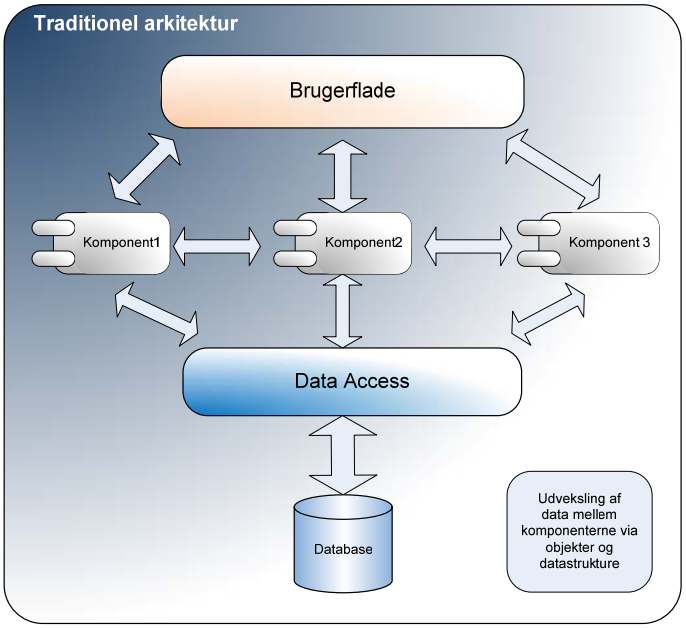
En service orienteret arkitektur (SOA) har til formål at skabe lav afhængighed mellem systemer og komponenter. SOA fokuserer på opbygningen af arkitekturen i det bagvedliggende system. SOA er ikke noget nyt koncept. Man har i mange år forsøgt at opnå lignende arkitekturer, eksempelvis gennem distribuerede objekter som DCOM eller CORBA og senest også første generation web services. SOA, som det er i dag, skal derfor mere ses som en evolution end en revolution.

4.1 TRADITIONEL ARKITEKTUR

Når der udvikles større systemer, er disse ofte designet og implementeret ud fra et antal komponenter, som oftest udviklet ud fra en objekt orienteret tilgang. Kommunikation og dataudveksling imellem disse baserer sig på klasser og datastrukturer. Metoden er anvendt i vid udstrækning og har også resulteret i mange veludførte udviklingsprojekter.

Ved at udvikle systemer på denne måde kan der dog opstå uhensigtsmæssigheder. Komponenter i sådanne systemer, der hver især realiserer noget forretningslogik, er meget tæt knyttet, da de udveksler data i form af objekter/klasser og ofte deler datastrukturer. Desuden arbejder alle systemets komponenter ofte i den samme database, som indeholder hele systemets data. [20]

Med den ofte anvendte objekt orienterede model kan systemet se ud som i Figur 21.



Figur 21 – Traditionel arkitektur

Figuren viser, at kommunikationen mellem komponenter, dataaccesslag og brugerfladen er baseret på objekter og strukturer. Dette giver en stor afhængighed mellem de forskellige komponenter og lag i systemet. Desuden er det ofte besværligt, at tage en del ud af systemet og genbruge et andet sted, da der med stor sandsynlighed vil være afhængigheder af underliggende lag eller andre komponenter i systemet.

Denne model kan derfor ofte give et stort stykke arbejde i forbindelse med følgende situationer.

Opdatering

Når komponenter i systemet skal opdateres, eller fejlrettes, har det ofte indflydelse på driften af hele systemet. Resultatet er ofte, at systemet helt eller delvist skal tages ud af drift, når opdateringerne udrulles. Dette vil være til gene for brugerne af systemet, som ikke vil være i stand til at anvende systemet overhovedet, selvom der kun bliver opdateret på en lille del af systemet.

Genanvendelse

Det er ikke ukendt, at man har brug for at genbruge forretningslogik i andre applikationer eller systemer i en organisation. Man forsøger derfor ofte at lave systemerne så generiske som muligt, netop med henblik på genbrug af komponenter eller kode. Desværre ender systemerne ofte med at være så tæt koblet, at det ikke kan betale sig at genbruge kode eller komponenter, set i forhold til at kode den ønskede funktionalitet igen tilpasset til den nye situation.

Skalering

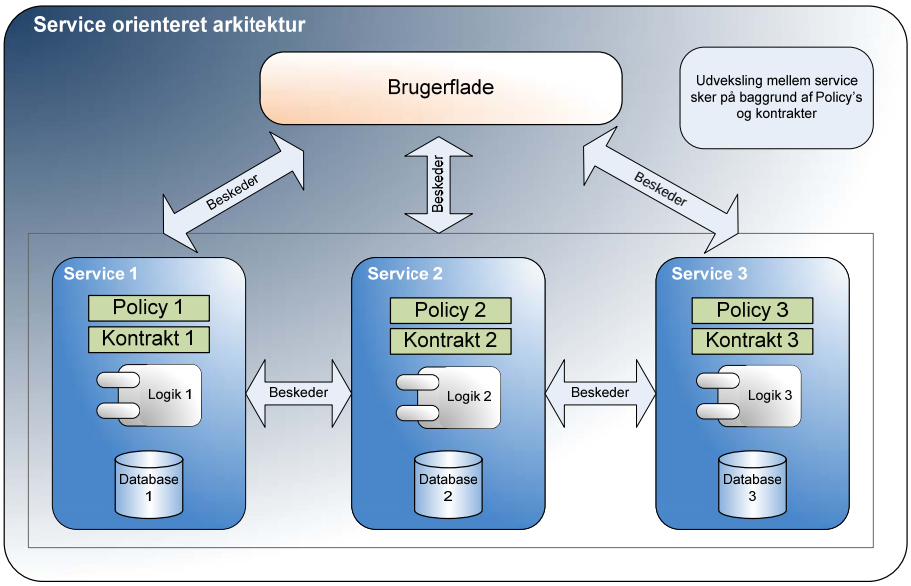
I tilfælde af at anvendelsen af systemerne stiger, og hardwaren ikke kan tilbyde den fornødne kraft, er det avanceret og tidskrævende at udvide hardware ressourcerne. Dette skyldes at systemerne er svære at lægge ud på flere maskiner, og man må ofte benytte sig af clustering eller lignende metoder. Dette vil i mange tilfælde være en administrativ og konfigurationsmæssig bekostelig affære.

Dette resulterer i, at man i mange virksomheder i dag kæmper med store komplekse arkitekturer, som har udviklet sig over en længere årrække. Disse systemer baserer sig ofte på grund af dette, også på mange forskellige teknologier. Dette betyder, at det vil kræve stor kompetence og ikke mindst mange økonomiske midler at skifte til anden teknologi. Samtidig er de eksisterende systemer dyre at vedligeholde på grund af deres kompleksitet.

4.2 SERVICE ORIENTERET ARKITEKTUR

SOA bygger på, at systemet opdeles i services på baggrund af forretningslogikken. Disse services opfylder hver især et funktionalitetsmæssigt behov. Dette er ikke så forskelligt fra tankegangen omkring komponenter i en objekt-orienteret design. Forskellen ligger i grænsefladerne mellem disse services. [1], [20]

Med en service orienteret arkitektur vil et system tilsvarende det i Figur 21 kunne modelleres som vist i Figur 22



Figur 22 – Service Orienteret Arkitektur

Med denne model er alle dele af systemet pakket ind i services med veldefinerede grænser. Hver service har sine egen data og må ikke være afhængig af andre end sig selv. Al kommunikation mellem services er i form af beskeder, der udveksles ud fra veldefinerede policies. Operationer, en service tilbyder, er defineret i kontrakter. For at man kan kalder et design service orienteret skal det opfylde nogle særlige principper. Disse principper eller hovedregler udstikker retningslinier for, hvordan en service bør fungere for at opnå den egenskab at kunne undgå de tidligere omtalte problemer, der findes i tæt koblede systemer. [15]

4.3 DE FIRE HOVEDREGLER[2]

4.3.1 HOVEDREGEL 1: GRÆNSER ER EKSPLICITTE

Grænsefalden i en service adskiller, og skjuler servicens interne logik fra omverdenen. Omverdenen skal ikke have kendskab til, hvordan servicen ser ud på den anden side af denne grænse, men udelukkede have kendskab til grænsefladen.

For klienten ser grænsefladen til en service identisk ud, om den kører på den samme maskine, på det lokale netværk eller på en server i et andet land. Klienten kan og skal ikke skelne imellem, hvor en service er fysik placeret. Dette giver mulighed for en høj fleksibilitet, men betyder ligeledes, at det kan være kosteligt at krydse grænsen til en service. Det er uforudsigeligt hvilket konsekvenser dette kan have i sikkerhedsrutiner eller i form af forsinkelser i netværk.

Formålet med denne regel, er at gøre udviklerne af services og klienter opmærksomme på, at mens services giver en høj fleksibilitet skal hvert kald til en service overvejes, da konsekvenserne af hvert kald ikke kan forudses.

De vigtigste punkter fra hovedregel 1:

- Hver kontakt med en anden service anses for at krydse en grænse.
- At krydse en grænse har konsekvenser.

4.3.2 HOVEDREGEL 2: SERVICES ER AUTONOME

En service må ikke have et styrende organ, den skal være styret, idriftsat og versioneret individuelt. Servicen må ikke have en afhængighed af andre, der kan resultere i at den går ned.

Servicen må godt benytte andre services, men skal kunne håndtere, hvis disse services ikke er tilgængelige og opretholde egen eksistens på trods af dette. Hvis en service ikke kan udføre sin opgave, fordi andre services den benytter, er utilgængelige, skal den kunne håndtere dette, og om nødvendigt informere sin klient om, at den ikke kan udføre sin opgave.

De vigtige punkter fra hovedregel 2:

- Der er ingen kontrollerende autoritet
- Services er idriftsat, styret og versioneret individuelt.
- Services må gerne benytte andre service, men skal kunne håndtere hvis disse fejler.'

[2] [1],[10],[20]

4.3.3 HOVEDREGEL 3: SERVICES DELER SKEMA OG KONTRAKTER IKKE OBJEKTER OG KLASSER

Denne regel viser en af de markante forskelle mellem service orienteret arkitektur og traditionel objekt orienteret arkitektur. Services udveksler ikke objekter og klasser, men skemaer og kontrakter.

Ved at holde sig til veldefinerede kontrakter og skemaer, baseret på XML, opnås platform uafhængighed og løs kobling mellem service og klient.

Kontrakter og skemaer bør forblive stabile. Klienter benytter disse til at kommunikere med servicen, og ændringer kan betyde, at klienter skal ændre deres opfattelse af servicen. Hvilket betyder ændringer i deres implementering.

Som alt andet udvikler services sig, hvilket til sidst betyder, at kontrakter og skemaer må ændres. Når dette gøres kræves det at der versioneres således, at servicen fortsat kan benyttes som tidligere ud fra de gamle kontrakter og skemaer.

De vigtige punkter for Hovedregel 3:

- Services interagerer ved ud fra kontrakter og skemaer.
- Kontrakter og skemaer bør forblive stabile over tid, hvis ikke skal de nye versioner altid være bagud kompatible.
-

4.3.4 HOVEDREGEL 4: KOMPATIBILITET ER BASERET PÅ POLICIES

Kompatibilitet er baseret på policies, der beskriver hvordan en service benyttes. Dette kan eksempelvis være hvilke sikkerhedsmæssige krav, der er defineret i forbindelse med kommunikationen, samt med hvilken protokol og hvor en service kan kontaktes. På den måde skabes den ønskede kompatibilitet mellem services.

Policies gør det således muligt, nemt at flytte en service fra et miljø til et andet. Der kan nemt ændres i servicens policy til at afspejle det nye miljøs krav.

De vigtige punkter for hovedregel 4:

- Krav for at kommunikere med en service defineres vha. policies.
- Ved at ændre policy'en kan der nemt ændres hvorledes der kommunikeres med servicen.

4.4 KARAKTERISTIKA VED SO/SOA

Som opsummering af de foregående afsnit, følger her en gennemgang af karakteristika ved implementering af en service orienteret arkitektur.

Services er isolerede

Ved at indkapsle forretningslogikken i services og gøre disse autonome, opnår man en stor uafhængighed. Denne uafhængighed bibringer, at man nemt kan ændre og opdater en enkelt service uden at det påvirker resten af systemet.

Services er placerings uafhængige

Tilgangen til en service er identisk hvad enten servicen bliver hosted på en maskine lokalt eller den hostes et andet sted i verden.

Services er platforms uafhængige

Hvis en service implementeres efter de gældende standarder og specifikationer kan der opnås platforms uafhængig. Dette giver en stor forretningsmæssig fordel. Dette medfører, at man har mulighed for at tilbyde sin service til et væsentligt bredere publikum, end hvis man arbejder med platformsafhængig kode.

Optimal for kodegenbrug

Med anvendelse af SOA er muligheden for kode genbrug kraftigt forbedret. Med SOA har man mulighed for at lægge ofte anvendt forretningslogik ud i selvstændige services, som kan bruges på tværs af applikationer.

Services er scalerings-venlige

Er systemet bygget op omkring en service orienteret arkitektur kan man indføre en directory-service, som applikationer eller brugere af systemet kan spørge om servere og adresser, de pågældende services er idriftsat på. I tilfælde af at antallet af brugere på systemet stiger, og det er en nødvendighed at ændre hardware konfigurationen, er det forholdsvis simpelt at sprede systemets services på flere servere og blot opdatere directory-servicen. Systemet vil for brugeren være uændret.

4.5 WINDOWS COMMUNICATION FOUNDATION (WCF)

Windows Communication Foundation (WCF) er Microsofts nye programmerings model til kommunikation i og mellem systemer. WCF er en del af det nye .NET 3.0 framework. .NET 3.0 er en kombination af det der tidligere hed *WinFX* og .NET 2.0. WCF var en del af WinFX og er nu en del af .NET 3.0. WinFX tilføjer de følgende 4 teknologier til .NET 2.0.

- *Windows Presentation Foundation* – Tidligere "Avalon" er et brugerflade API som er baseret på .NET, XML, vector grafik og gør brug af 3D hardware acceleration.
- *Windows Workflow Foundation*– Ny teknologi til at definere, administrere og eksekvere arbejdsgange.
- *Cardspace* – Sikker opbevaring og benyttelse af digitale identiteter.
- *Windows Communication Foundation* – Tidligere "Indigo"

4.5.1 GRUNDLAGET FOR WINDOWS COMMUNCATION FOUNDATION

Der findes en række teknologier fra Microsoft, der kan benyttes til at bygge distribuerede systemer. Disse er *ASP.NET web services, Web Services Enhancements.*, *Microsoft Message Queuing, Enterprise Services/COM+* og *.NET Remoting.* [9], [6]

Microsoft samler nu disse og en række nye teknologier i WCF, således at udviklere har én programmerings model hvor al kommunikation er samlet. WCF er udviklet med en række helt specifikke mål for øje. To af disse vil det efterfølgende blive forklaret nærmere.[6]

4.5.1.1 Kompatibilitet på tværs af platforme og leverandører[3]

WCF benytter standarderne *WSDL* til beskrivelse af services og *SOAP* beskeder til kommunikationen mellem services og klienter. WCF understøtter ligeledes *WS-I Basic Profile* specifikationen, hvilket betyder, at WCF kan kommunikere med andre ikke WCF systemer, hvis disse ligeledes understøtter WS-I Basic Profile. Basic Profile definere hvorledes SOAP, WSDL og UDDI skal benyttes, således en web service overholder Basic Profile specifikationerne.

WSDL – Web Service Description Language specificerer hvorledes en web services beskrives, således at klienter kan kommunikere med den. Beskrivelsen indeholder bla. information om hvilken protokol, besked format og hvilke operationer servicen understøtter. [16]

UDDI – Universal Description, Discovery and Integration er en "telefonbog" for services hvori web services kan registreres, således det er mulig at finde og benytte web services som andre har lavet og stiller til rådighed.[22]

SOAP er en protokol til udveksling af beskeder, SOAP er transport neutral, dvs. den er uafhængig af den underliggende transport metode. Typisk benyttes HTTP, men f.eks. SMTP eller FTP kan ligeledes benyttes. [21]

Udover WS-I Basic Profile understøtter WCF også en række af WS-* specifikationerne der er opstillet af bla. Microsoft, SAP og IBM og som udvider WS-I Basic Profile specifikationerne med bla. nye muligheder inden for sikkerhed, troværdighed og transaktioner. [19], [6] Der kan læses mere om WS-* specifikationerne i Bilag 5

[3] [6],[1]

Da systemet, der udvikles i forbindelse med projektet, kun afdækker en del af et kundekontaktcenters behov, er det en fordel at systemet arbejder med en række standarder at specifikationer, som er accepteret af en lang række leverandører, således det er muligt og integrerer med eksisterende systemer i kundekontaktcenteret, hvis der skulle opstå disse behov.

4.5.1.2 Valgfri kommunikations teknologi.[4]

Et problem med de nuværende teknologier er, at der tidligt i udviklingsforløbet skal tages stilling til hvilke teknologier der skal benyttes, da teknologivalget vil have indflydelse på hvorledes forretningslogikken i systemet implementeres. Dette betyder ofte, at hvis der ønskes at skifte fra én kommunikationsteknologi til en anden, skal der foretages betydelige ændringer i forretningslogikkens implementering eller der skal laves et mellemled der understøttede den nye ønskede kommunikations teknologi.

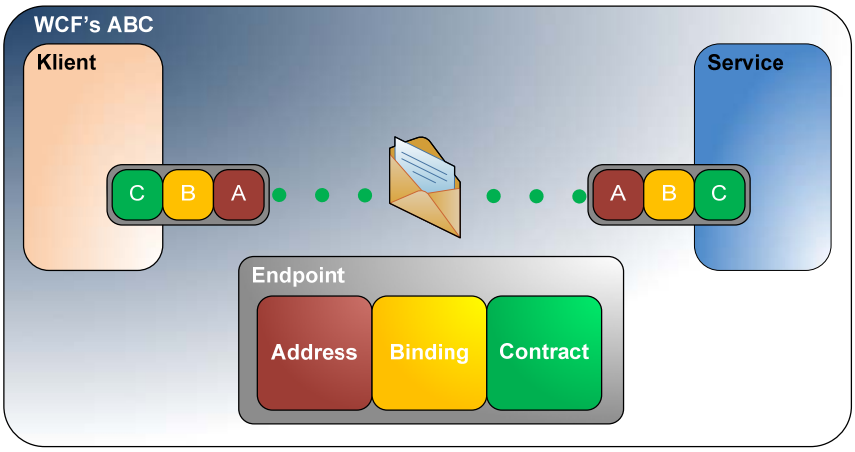
WCF isolerer forretningslogikken fra kommunikationsteknologien, så det evt. skift af kommunikations teknologi ikke har indflydelse på forretningslogikken.

Grunden til dette er, at der med WCF kun skal ændres i konfigurationsfiler. I en ikke-WCF løsning skal der typisk implementeres et nyt lag som understøtter den nye teknologi eller direkte laves ændringer i forretningslogikken.

I forbindelse med projektet, er det en fordel, at forretningslogikken kan implementeres uden at tage stilling til, hvorledes kommunikationen med denne skal foregå. Det betyder, at der opnås en høj fleksibilitet i systemet, således at det er nemt at omstrukturere systemet til nye behov, f.eks. hvis det bliver relevant at integrere med andre kundekontaktcenter systemer.

4.5.2 WCF'S ABC - ADDRESS, BINDING, CONTRACT

I Figur 23 er illustreret hvorledes en klient kommunikere med en WCF service, og hvilket elementer der indgår.



Figur 23 – Windows Communication Foundations ABC

En service skal minimum have ét *endpoint,* for at det er muligt at tilgå denne. Et endpoint består af adresse, binding og kontrakt. Adressen beskriver hvor servicen kan findes, bindingen hvordan der kommunikeres med den og kontrakten hvad den tilbyder af funktionalitet. [1],[7],[8]

[4] [6],[1]

4.5.2.1 Kontrakter

Kontrakter i WCF definerer hvorledes en klient kan benytte en service. Der er 3 typer kontrakter i WCF: Service, Data og Message. Service kontrakten beskriver hvilke operationer servicen kan udføre, mens data kontrakter beskriver data strukturer, som evt. benyttes til disse operationer. Message kontrakter kan benyttes til at styre strukturen af de SOAP beskeder, som service og klient udveksler. [1]

4.5.2.1.1 Service kontrakter

Service kontrakter beskriver hvad en service kan ud fra en liste af *service operationer*. En service operation minder i høj grad om almindelige metoder. De kan modtage parametre og returnere værdier præcis som en metode. [8]

Eksempel 1 – Service kontrakt

```
[ServiceContract(Namespace="http://Ciw.CiwService.Questionaire")]
public interface IQuestionaire
{
   [OperationContract]
   bool CreateQuestionaire(dcQuestionaire newQuestionaire);

   [OperationContract]
   dcQuestionaire[] GetAssignedQuestionairies(string username, bool bIncludeSubData);
   …
   …

   [OperationContract]
   dcReply GetReply(string username, Guid QuestionaireIdentification);
}
```

I Eksempel 1 ses en servicekontrakt i WCF. Koden er som et almindeligt interface, med den forskel, at interfacet er dekoreret med [ServiceContract] og hver metode med [OperationContract]. På denne måde ved WCF, at der her er tale som en servicekontrakt med dens operationer. Det er ikke nødvendigt at benytte interfaces. Man kan dekorere en klasse på samme måde og definere en service kontrakt således. Dekorationerne benyttes af WCF til at genere WSDL beskrivelsen af servicen.

Eksempel 2 – Service implementering

```
[ServiceBehavior]
public class Questionaire: IQuestionaire
{
   [OperationBehavior]
   public bool CreateQuestionaire(dcQuestionaire newQuestionaire)
   { … }

   [OperationBehavior]
   public dcQuestionaire[]GetAssignedQuestionairies(string username,bool bIn…)
   { … }
   …
   …
   [OperationBehavior]
   public dcReply GetReply(string username, Guid QuestionaireIdentification)
   { … }
}
```

Implementering af servicen gøres herefter i en almindelig klasse, der implementerer interfacet og dekoreres med [ServiceBehavior] og [OperationBehavior]. Servicekontrakter kan dekoreres med yderligere attributter, der gør det muligt at definere krav til bindings type, hvorvidt sessioner er krævet mm. Hvis disse krav ikke opfyldes, nægter servicen at starte, så det ikke er muligt f.eks. at eksponere en service, der kræver sikkerhed i bindingen, men er konfigureret med en binding uden sikkerhed.

4.5.2.1.2 Data kontrakter

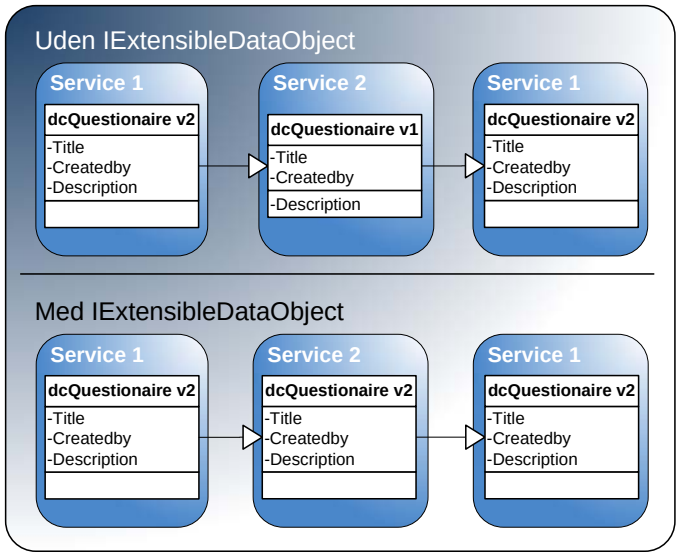
Datakontrakter kan benyttes til at overføre egen definerede datastrukturer mellem klient og service. Der benyttes data kontrakter i stedet for objekter til at overfører data strukturere, således det er muligt at overholde den 3. hovedregel for service orientering, se afsnit 4.3.3. Et simpelt eksempel på en data kontrakt ses herunder. En datakontrakt har samme struktur som en almindelig klasse, der blot er dekoreret med [DataContract] og [DataMember]. DataContract og DataMember dekorationerne tilføjes, så WCF kan serialize klassen til et XML skema (XSD) der medtages i WSDL beskrivelsen af servicen. [1],[6],[8]

Eksempel 3 – Datakontrakt

```
[DataContract(Namespace = "http://Ciw.CiwService.Questionaire")]
public class dcQuestionaire : IExtensibleDataObject
{
   [DataMember(IsRequired=true)]
   public string Title;
   [DataMember(IsRequired=false)]
   public string Description;
   …
   …
   [DataMember(IsRequired=true)]
   public string CreatedBy;

}
```

Som med servicekontrakter er det i datakontrakter muligt at bruge yderligere dekorationer blandt andet *IsRequired,* der angiver om datamemberet kan udelades. I Eksempel 3 er "Description" ikke krævet, hvilket betyder, at en evt. klient kan udelade det og stadig benytte servicen og datakontrakten. Dette muliggør versionering af datakontrakter, da der i senere versioner af kontrakten kan tilføjes datamembers med IsRequired = false uden at kontrakten brydes, således at klienter, der benytter den gamle kontrakt, kan fortsætte med at benytte servicen.

Som det ses af Eksempel 3 implementerer klassen, som udgør datakontrakten, interfacet *IExtensibleDataObject*. Dette gøres for at understøtte "*Round-tripping*". Round-tripping er betegnelsen for når data passerer fra en ny version til en gammel og tilbage til en ny version af en datakontrakt. Hvis ikke datakontrakten implementerede IExtensibleDataObject ville Description gå tabt, når dataene passerede den gamle version af datakontrakten i f.eks. en service, der benyttede den gamle kontrakt. [11]



Figur 24 – IExtensibleDataObject

Når IExtensibleDataObject er implementeret, gemmes Description i et *ExtensionData* element. Når dataene igen når en service, der benytter den nye version af kontrakten, lægges Description tilbage i dets datamember. Dette er illustreret i Figur 24

4.5.2.1.3 Beskedkontrakter

Når WCF laver en SOAP besked ud fra de servicekontrakter og datakontrakter, der er defineret, styrer den selv hvorvidt elementer skal indgå i header eller body delen af SOAP beskeden. Med beskedkontrakter er det muligt, at definere hvor i beskeden bestemt information skal placeres. Formålet med dette er primært kompatibilitet med andre systemer, således det er muligt at opfylde deres krav til strukturen af SOAP beskeder. [1]

Der vil i projektet blive gjort brug af servicekontrakter og datakontrakter, men da der ikke pt. skal arbejdes sammen med andre systemer, vil det ikke blive relevant at benytte beskedkontrakter, for at opfylde deres krav til strukturen af SOAP beskeder.

4.5.2.2 Binding

Binding i en WCF service definerer en række ting i forbindelse med, hvordan klienter kommunikerer med servicen. Her i blandt hvilken transport metode der benyttes, hvordan beskeder er kodet, sikkerhed, sessioner og transaktioner. [1],[8]

Standard er der 9 binding typer i WCF. Et udvalg af disse kan ses i Tabel 3, resten kan ses i bilag 7. Hvis ikke en af disse er passende, er det ligeledes muligt at implementere sin egen type, således det teoretisk set er muligt at lave en binding der benytter f.eks. SMTP som transport metode.

Tabel 3 – Standart binding typer i Windows Communication Foundation

Binding	Kompatiblitet	Sikkerhed	Session	Transaktioner	Duplex
BasicHttpBinding	Basic Profile 1.1	Ingen, Transport, besked,	Ingen	None	Nej
WSHttpBinding	WS-*	Transport, Besked, Mixed	Ingen, Transport, Reliable Session	None, Yes	Nej
NetTcpBinding	.NET	Transport, Besked	Reliable Session, Transport	None, Yes	Ja
NetNamedPipeBinding	.NET	Transport	Ingen, Transport	None, Yes	Ja

[11]

Af kompatibilitetskolonnen fremgår det på hvilket niveau bindingen er kompatibel. Der er Basic Profile, WS-* og .NET. Som det fremgår af kapitel 4.5.1.1 er Basic Profile og WS-* web service standarder. NetTcpBinding og NetNamedPipeBinding er kun kompatible med andre .NET/WCF services bla. fordi de benytter binær kodning af beskeder i stedet for XML. Hvis kompatiblitet med andre platforme end WCF ikke er relevant, er det en fordel at benytte en af disse *bindings*, da binær kodning giver en mindre besked størrelse og derfor en højere hastighed. Hvis service og klient kører på samme maskine, vil det ligeledes være en fordel at benytte NetNamedPipeBinding, da dette giver en højere hastighed end de http/tcp baseret bindinger.

I projektet vil der som udgangspunkt blive gjort brug af de i Tabel 3 viste bindinger, da disse dækker behovene for transport og besked sikkerhed. De yderligere bindinger, som kan ses i bilag 7, forventes ikke vil blive brugt, bla. fordi der ikke er behov for Peer-to-peer eller Microsoft Message Queue funktionalitet.

Session, Transaktioner og duplex vil ikke blive benyttet i projektet, og der vil derfor ikke blive gået yderligere i detaljer om dette.

4.5.2.3 Adresse

En service skal definere mindst én adresse, dette kaldes *baseaddress*. Udover denne kan der defineres en eller flere endpoint adresser. Disse kan defineres relativt i forhold til base adressen eller uafhængigt. [1]

En service kunne f.eks. have base adressen *http://iceman.nextcar.dk:8080/CarService* og et endpoint på den relative adressen "Tires" hvilket giver endpointet den fulde adresse: *http://iceman.nextcar.dk:8080/CarService/Tires*. Samme service kunne ligeledes have et named pipe endpoint med adressen: *net.pipe://localhost/Tires*.

En adresse består af et præfix der identificerer hvilken transport metode der benyttes, f.eks. "http", et maskinenavn, f.eks. "iceman.nextcar.dk", et portnummer og en sti f.eks, "/CarService/". Hvis transport metoden er named pipes skal der ikke angives et portnummer.

4.5.3 SIKKERHED[5]

Sikkerhed er et vitalt punkt i alle systemer, men i systemer, hvor services evt. er eksponeret på Internettet og alle har mulighed for at forsøge at tilgå disse services, er det essentielt, at sikkerhedsmekanismer er på plads.

Sikkerhed kan opdeles i de 4 følgende punkter:

- **Validering:** Er en entitet, typisk en bruger, hvem han udgiver sig for at være. Entiteten præsenterer legitimation i form af f.eks. certifikater eller en brugernavn, password kombination.
- **Autorisering:** Er en entitet autoriseret til at udføre et given handling.
- **Besked integritet:** Forsikring om at beskeder mellem to parter ikke bliver ændret undervejs.
- **Besked fortrolighed:** Forsikring om at det kun er den tiltænkte modtager, der kan læse beskeden.

Grundlæggende er der 2 muligheder for sikkerhed i WCF, transport og besked. Forskellen på de to typer er grundlæggende, at transport benytter en beskyttet "tunnel" f.eks. SSL til, at sikre beskeder mellem to parter. Besked sikkerhed beskytter i stedet den enkelte besked, uafhængigt af transport metoden.

Transport sikkerhed benytter den underliggende transport metode, til at opfylde de 4 sikkerhedspunkter, dette kunne f.eks. være HTTPS. Til validering og autorisering er transport sikkerhed begrænset til, at benytte de legitimationstyper som den underliggende transport metode understøtter. I Tabel 4 ses hvilket typer der er understøttet når transport metoden er https.

Besked sikkerhed benytter WS-Security fra WS-* specifikationerne og har den fordel, at den er uafhængig af transport typen, hvilket betyder, der er mulighed for flere legitimationstyper til validering og autorisering, disse ses i Tabel 4. [11]

Tabel 4 – Legitimationstyper ved forskellige transporttyper i WCF

Transport metode	**Legitimationstyper**	
	Transport sikkerhed	**Message sikkerhed**
HTTP	Ingen, Basic, Digest, Ntlm, Windows, Certifikat	Ingen, Windows, Token, Username, Certifikat
TCP	Ingen, Windows, Certifikat	Ingen, Windows, Token, Username, Certifikat
Named pipe	Windows	Understøttes ikke

I projektet ønskes det, at benytte et brugernavn/password kombination som legitimationstype. Ligeledes ønskes det ikke at validere brugere imod et Windows Active Directory, hvorfor Basic, Digest og Windows ikke er brugbare, da de alle benytter Windows Active Directory. Dette betyder, der vil blive benyttet besked sikkerhed, hvilket giver muligheden for at validere brugere vha. egen udviklede sikkerhedsmekanismer.

[5] [1],[7]

4.5.4 BESKED UDVEKSLING

WCF understøtter som standard 3 typer beskedudveksling. [1]

- **Simplex** – Envejs kommunikation hvor en klient sender beskeder til en service uden at der kommer svar retur. Services, der benytter simplex, sender ikke selv nogen beskeder, men modtager kun.

  ```
  [OperationContract(IsOneWay=true)]
  void ReportLapTime(DateTime LapTime);
  ```

  [OperationContract] elementet dekoreres yderligere med *IsOneWay=true* og metoden skal returnere void for at den fungerer som simplex.

- **Duplex** – Tovejs kommunikation uden synkronisering, dvs. de to parter kan sende beskeder til hinanden uden at vente på svar for her sendt besked.

  ```
  [ServiceContract(CallbackContract=typeof(ICarDuplexContract)]
  public interface ICarService
  {...}
  ```

  Servicekontrakten dekoreres med *CallBackContract* som angiver hvilken kontrakt der skal benyttes, når servicen skal sende en besked til den modsatte part. Ikke alle bindings typer understøtter duplex, hvilke kan ses i tabellen i bilag 7.

  - **Request-reply** – Synkron tovejs kommunikation, dvs. klienten venter på svar fra servicen inden den foretager sig yderligere. Modsat Duplex, så kan servicen ikke sende beskeder til klienten på nær reply på beskeder. Eksempel 1 viser en service med Request-reply operationer. Request-reply er standard type for besked udveksling, da denne benyttes medmindre service eller operationer er dekoreret som Simplex eller Duplex.

I projektet vil der blive benyttet request-reply kommunikation, da dette er den mest simple at benytte, og der ikke er behov for at benytte andre typer for at implementere systemet.

4.5.5 HOSTING

WCF understøtter grundlæggende 2 forskellige typer hosting. [1]

- ***Self-hosting:*** Self-hosting dækker over, når en service hostes i f.eks. en konsol eller Windows applikation. Ved self-hosting er udvikleren ansvarlig for servicens "helbred" og skal implementere funktionalitet til at håndtere evt. "helbredsproblemer" som f.eks. en genstart af servicen.

- ***IIS-hosting:*** Ved IIS-hosting hostes servicen af en Internet Information Services (IIS) web-server. Ved IIS-hosting er det ikke nødvendigt at starte servicen. IIS serveren sørger for at starte servicen når der er behov for det, dvs. når der kommer indgående beskeder til den. IIS overvåger ligeledes servicens helbred og foretager selv genstart af servicen, hvis der opstår helbredsproblemer.

Under udviklingen vil services blive self-hosted i konsol applikationer, da dette er nemmere at administrere i et udviklingsmiljø hvor services ofte skal genstartes pga. ændringer i implementeringen. Det endelige system vil blive hosted på IIS servere da dette giver en robust hosting uden og skulle implementere yderligere. I bilag 9 kan ses et simpelt eksempel på en implementeret konsolhost.

4.6 RESUME

Der er blevet forklaret hvad en service orienteret arkitektur dækker over, og hvorledes den adskiller sig fra tidligere benyttede arkitekturer. De fire hovedregler for service orienteret arkitektur er beskrevet. Disse hovedregler vil indgå som retningslinier for designet af systemets arkitektur i det følgende kapitel.

De grundlæggende dele af Windows Communication Foundation er gennemgået, med henblik på hvorledes de vil blive benyttet i systemet.

Kapitel 5

DESIGN

På baggrund af den tidligere udformede kravspecifikation i kapitel 2 og analyse i kapitel 3 vil der fokuseres på at designe en service orienteret arkitektur til systemet. I opbygningen af denne arkitektur vil den opnåede viden om Service Orientering og WCF indgå som et vigtigt element. Der vil desuden blive opstillet service og datakontrakter til de enkelte services.

Kapitlet vil desuden på baggrund af domæne modellerne fra kapitel 3 opstille database design til enkelte service.

5.1 BAGGRUND FOR DESIGN

I analysen i kapitel 3 er der defineret et system bestående af en rækkes services. Hver af disse services opfylder en række funktionelle og ikke funktionelle krav.

Det ønskes at opstille et design, der kan realisere dette system ud fra de stillede krav, men yderligere ønskes det at forsøge at realisere følgende:

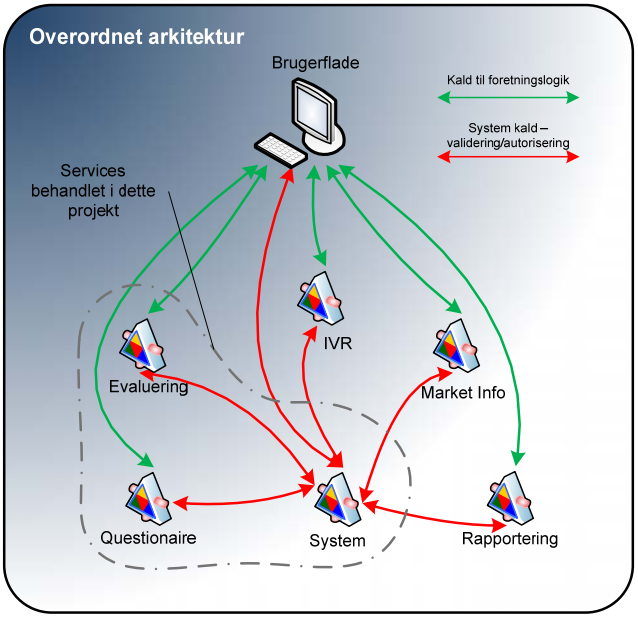
- Systemet skal være nemt at udvide med nye værktøjer i form af services, da det er et område der er i udvikling. Der kan hurtigt blive behov for nye værktøjer, som skal indgå i systemet.

- Da der ofte er forskellige behov for, hvilke værktøjer en virksomhed har brug for, skal det være muligt, at tilrette systemet til at indeholde den delmængde af services, det pågældende kundekontaktcenter har behov for.

Disse punkter er affødt af kapitel 1 afsnit 1.6 og er ikke en del af kravspecifikationen. Det ønskes dog at medtage disse punkter i designet af systemet, for jf. problemformuleringen at undersøge, om anvendelsen af en service orienteret arkitektur kan bidrage med mere end blot at realisere de til systemet opstillede krav.

5.2 OVERORDNET ARKITEKTUR

På baggrund af analysen i kapitel 3, den grundlæggende viden om SOA og WCF fra kapitel 4 og ønskerne fra forrige afsnit er der opstillet følgende overordnede struktur.

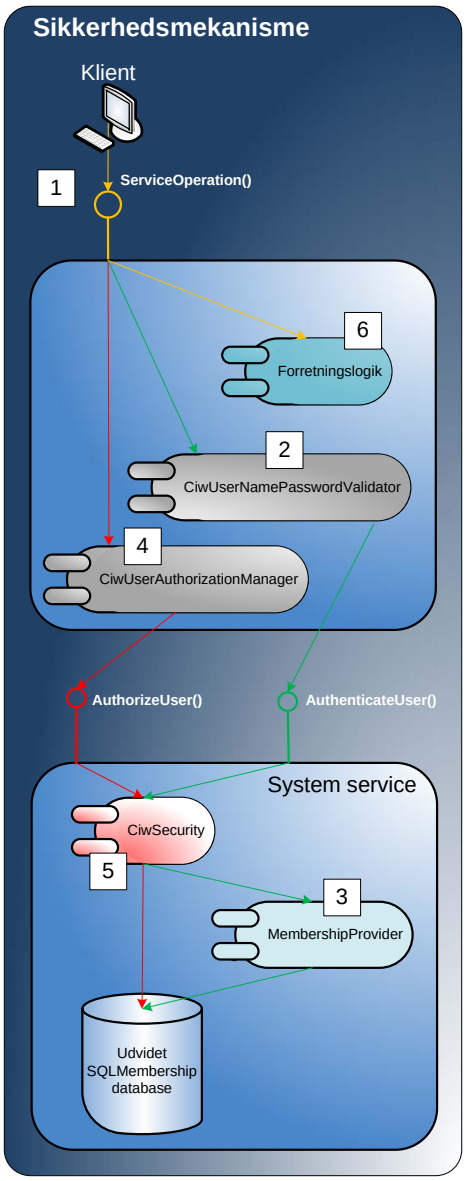


Figur 25 – Overordnet arkitektur

I Figur 25 er vist, hvorledes den overordnede struktur er designet. Det er illustreret, hvorledes de enkelte services benytter systemservicen til validering og autorisering af brugere. Desuden illustreres det, at systemservicen kommunikerer med de resterende services, når disse tilføjes systemet. Endelig kommunikerer brugerfladen med systemets services.

5.2.1 SIKKERHEDSMEKANISME

Figur 26 viser hvorledes sikkerhedsmekanismen i systemet er designet. I bilag 8 kan systemsekvensdiagrammet, der ligger til grund for sikkerhedsmekanismen, ses i Figur 26. Sikkerhedsmekanismen skal benyttes af services i systemet til at foretage validering og autorisering.



Figur 26 – Sikkerhedsstruktur

Ved punkt 1 kalder en klient en operation i en service. Servicen benytter *CiwUserNamePasswordValidator* til at validere klienten ud fra brugernavn og password.(Punkt 2.)

CiwUserNamePasswordValidator benytter herefter *AuthenticateUser* operationen i Systemservicen. *AuthenticateUser* er implementeret i CiwSecurity. CiwSecurity benytter .NET's *SqlMembershipProvider* til at validere brugeren. SqlMembershipProvideren slår op i databasen for at forsøge at validere brugernavn og password. Hvis valideringen ikke er succesfuld logges dette i systemet.

Hvis valideringen er succesfuld, fortsættes der ved punkt 4 med at undersøge, om klienten har autorisation til at udføre operationen. Servicen benytter *CiwUserAuthorizationManager* til at autorisere klienter.

CiwUserAuthorizationManager benytter systemservicens *AuthorizeUser,* der er implementeret i CiwSecurity (Punkt 5). *AuthorizeUser* laver et opslag i databasen for at undersøge, om klienten er autoriseret til at udføre operationen.

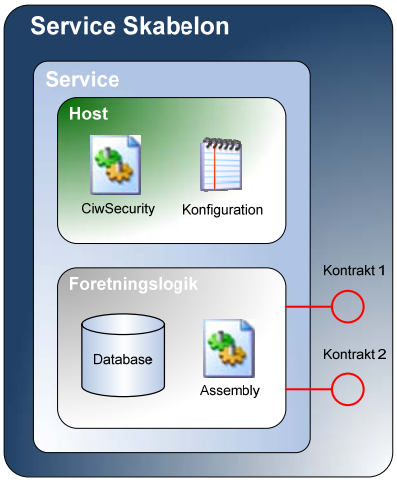
Hvis klienten er autoriseret, udføres operationen i forretningslogikken i punkt 6. Hvis klienten ikke er autoriseret, logges det mislykkede forsøg i systemet.

I Figur 26 angiver den grå farve hvilke dele af strukturen, der kan udskiftes ved ændring af konfigurations filerne til servicerne. I servicen er det muligt at skifte fra *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* til standard WCF validerings og autorisations metoderd hvis dette ønskes. Servicen har derfor ikke en stærk binding til systemservicen, hvilket også er en af SOA's hovedregler.

5.2.2 SERVICE SKABELON

For at gøre tilgangen til at udvikle services til systemet enkel er der defineret en service skabelon. Skabelonen kan bruges som udgangspunkt for en ny service i systemet og indeholder de nødvendige elementer for at indgå i systemet. Det er ikke et krav, at services er opbygget efter denne skabelon for at indgå i systemet. Skabelonen er blot en hjælp til at implementere nye services.

Grunddesignet for en service er skitseret i Figur 27. Skabelonen består af to hovedelementer: Host og forretningslogik. Disse består ligeledes af to elementer som vist i figuren.



Figur 27 – Service skabelon

- Host
  - CiwSecurity
  - Konfiguration
- Forretningslogik
  - Assembly med forretningslogik
  - Database

I Figur 27 er der vist en service bestående af de elementer nævnt ovenfor, som tilsammen udgør en service.

I bunden ligger forretningslogikken. Forretningslogikken består af en assembly med den implementerede forretningslogik og evt. tilhørende database. Assemblyen skal jf. afsnit 4.5.1.2 kun indeholde implementeringen af forretningslogikken.

Databasen anvendes kun af servicen selv. I figuren er databasen vist som om den er indkapslet i servicen. Dette er ikke tilfældet, men illustrere blot, at databasen tilhører servicen og kun bruges af denne

Hosten indeholder assemblyen CiwSecurity. CiwSecurity implementerer funktionalitet, således at servicen kan benytte sikkerhedsmekanismen beskrevet i forrige afsnit ved blot at opsætte dette i konfigurationen.

Hosten indeholder endvidere konfigurationen af servicen. I konfigurationen defineres servicens policy, dvs. hvorledes der kommunikeres med servicen, som beskrevet i afsnit 4.3.4.

5.2.2.1 Service assemblys

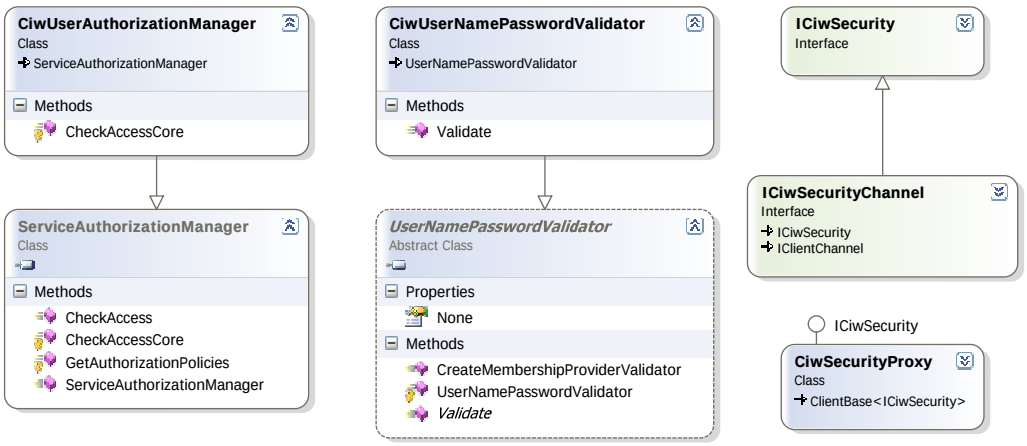
5.2.2.1.1 CiwSecurity

Services til systemet udvikles ud fra en service skabelonen. Til serviceskabelonen knytter der sig CiwSecurity assemblyen som implementerer validering og autorisering af brugeren imod systemservicen.

Til validering og autorisation benyttes klasserne *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager,* som er nedarvet fra WCF klasserne *UserNamePasswordValidator* og *ServiceAuthorizationManager*. Ved at nedarve fra disse klasser, kan *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* benyttes af WCF services til validering og autorisering i stedet for de standard mekanismer, WCF normalt gør brug af.

På denne måde er det muligt at få services til at validere og autorisere imod systemservicen ved blot at angive *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* som validerings og autoriserings mekanismer i konfigurationen af servicerne.

For at *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* kan kommunikere med systemservicen, er der inkluderet en proxy til systemservicen, som bla. indeholder kontrakten.

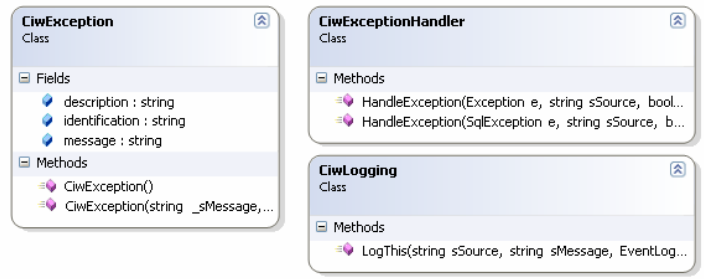


Figur 28 – Klassediagram for CiwSecurity

I Figur 28 ses klasserne i CiwSecurity. Her ses hvorledes validerings og autoriserings klasserne er nedarvet, og at CiwSecurity indeholder en proxyklasse og kontrakt til systemservicen.

5.2.2.1.2 CiwErrorHandling

Denne Assembly implementerer funktionalitet, der kan benyttes til fejlhåndtering og logning i forretningslogikken. De nødvendige klasser til denne assembly er vist i klasse diagrammet i Figur 29.



Figur 29 – Klasse diagram ErrorHandling assembly

Assemblyen indeholder en exceptionhandler klasse, som kan anvendes i forbindelse med fejlhåndtering i forretningslogikken. Assemblyen indeholder også en klasse der kan logge information hvis det er nødvendigt.

Da der fra service ikke bør kastes "almindelige" .NET exceptions til klienter, sørger *CiwExceptionHandler* for at pakke .NET exceptions ind i typen *FaultException<CiwException>* og kaste denne videre til klienten i stedet. [11]

5.3 SYSTEMSERVICE

Den overordnede arkitektur baserer sig på systemservicen som den, der håndterer brugeradministration, validering og autorisering. Samt håndtering af hvilke services der er tilknyttet systemet. Desuden håndtere den grupper, roller og deres rettigheder i systemet.

Til bruger og rolle håndtering anvendes .NET's SqlMembership- og SqlRoleProvider. Disse benyttes, da de implementerer en stor del af den funktionalitet, som systemservicen skal tilbyde. Det er derfor oplagt at tage udgangspunkt i dem frem for at implementere dette fra grunden. Databasen til systemservicen tager derfor udgangspunkt i den database, disse to providere benytter. Der er dog foretaget udvidelser af denne for at understøtte grupper, services, operationer og rettigheder til disse operationer.

Efterfølgende vil kontrakter, klasser og systemsekvensdiagrammer, der er opstillet for systemservicen, blive gennemgået.

5.3.1 KONTRAKTER

5.3.1.1 Servicekontrakter

I systemservicen er der opstillet følgende servicekontrakter:

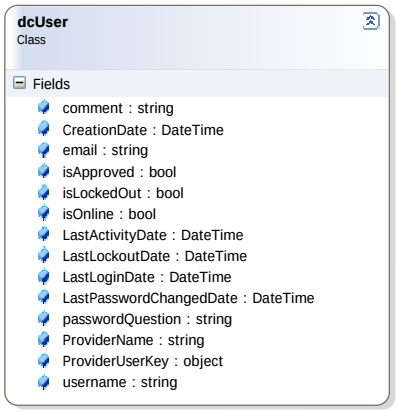
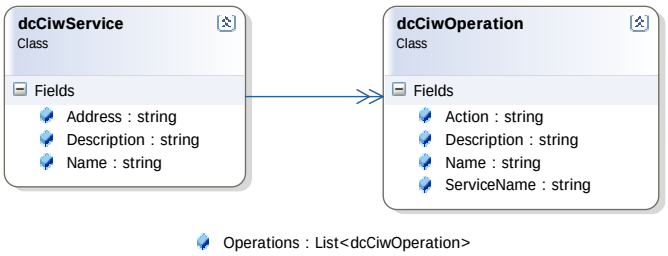
- **ICiwService** - Administration af services i systemet
- **IRoles** - Administration af roller i systemet
- **IGroups** - Administration af grupper i systemet
- **IUsers** - Administration af brugere i systemet
- **ICiwSecurity** - Validering og autorisering.

Systemservicen er opdelt i 5 kontrakter, en for hver funktionsområde. Dette er dels gjort for at gøre koden mere overskuelig og dels for at opnå en højere fleksibilitet bla. med hensyn til at give hver kontrakt deres eget *endpoint* som beskrevet i afsnit 4.5.2. Det er f.eks. nødvendigt at opsætte et separat *endpoint* til *ICiwSecurity,* da denne skal have andre sikkerheds indstillinger end de øvrige kontrakter. Dette skyldes at operationerne i *ICiwSecurity,* har med brugervalidering og autorisering at gøre. Disse skal tillade ikke validerede og ikke autoriserede klienter, da de netop benyttes til at validere og autorisere klienter.

5.3.1.2 Datakontrakter

I Systemservicen er der designet tre data kontrakter, Datakontrakterne bruges til at udveksle data med systemservicen. Datakontrakterne er udformet jf. afsnit 4.5.2.1.2

Tabel 5 – Datakontrakter i systemservicen

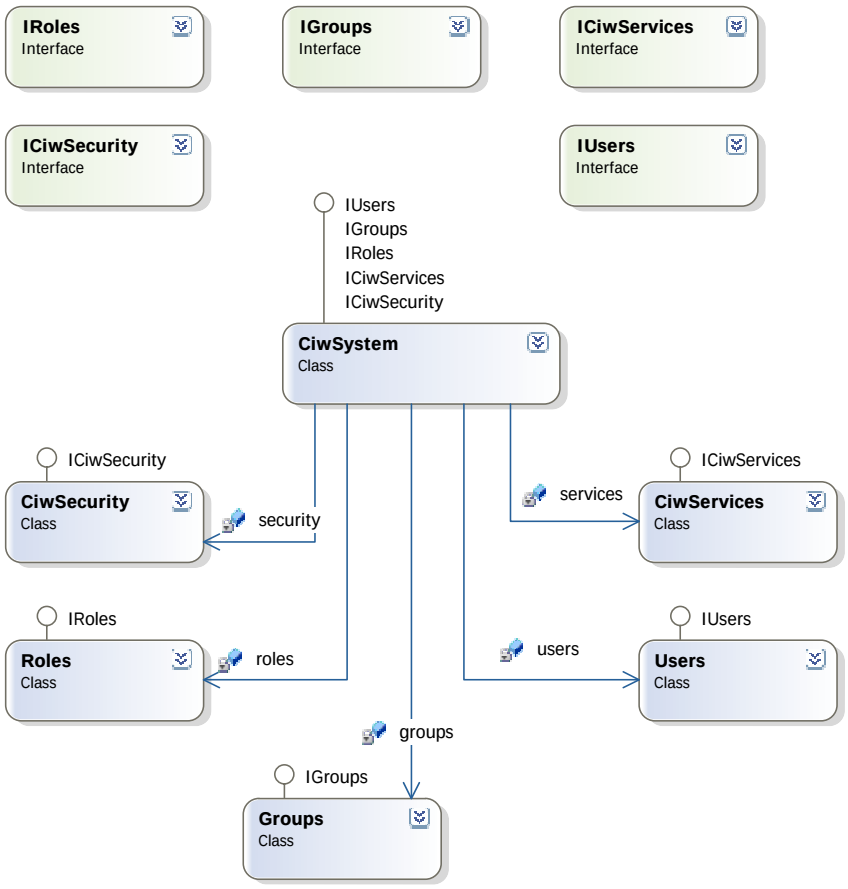
dcUser	dcCiwService	dcCiwOperation
dcUser kontrakten repræsenterer en bruger i systemet og benyttes til ind og uddata fra servicen, der omhandler brugere. Da *SqlMembershipprovideren* benyttes til brugerhåndtering, indeholde, dcUser mange af de samme felter som .NET klasse *MembershipUser,* som SqlMembershipProvideren benytter.	*dcCiwService* og *dcCiwOperation* datakontrakterne benyttes i forbindelse med udveksling af data omkring services. *dcCiwService* indeholder information om en service og en liste af *dcCiwOperation*. *dcCiwOperation* indeholder information om en serviceoperation.	
		

5.3.2 KLASSER I SYSTEM-SERVICEN

I tabellen herunder er listet de klasser, der er opstillet til systemservicen, samt en beskrivelse af hvad formålet er med klassen. Klassediagrammet for systemservicen er vist i Figur 30

Tabel 6 – Klasser i systemservicen

Navn	Beskrivelse
IUsers	Kontrakt til alle operationer der knytter sig til brugerhåndteringen
IRoles	Kontrakt til alle operationer der knytter sig til roller
IGroups	Kontrakt til alle operationer der knytter sig til grupper
ICiwSecurity	Kontrakt til operationer til validering og autorisering
iCiwServices	Kontrakt til operationer vedrørende service orientering
CiwSystem	Klasse der implementere alle servicekontrakterne
CiwSecurity	Klasse der implementerer ICiwSecurity funktionaliteten
CiwServices	Klasse der implementerer ICiwServices funktionaliteten
Users	Klasse der implementerer IUsers funktionaliteten
Roles	Klasse der implementerer IRoles funktionaliteten
Groups	Klasse der implementerer IGroups funktionaliteten

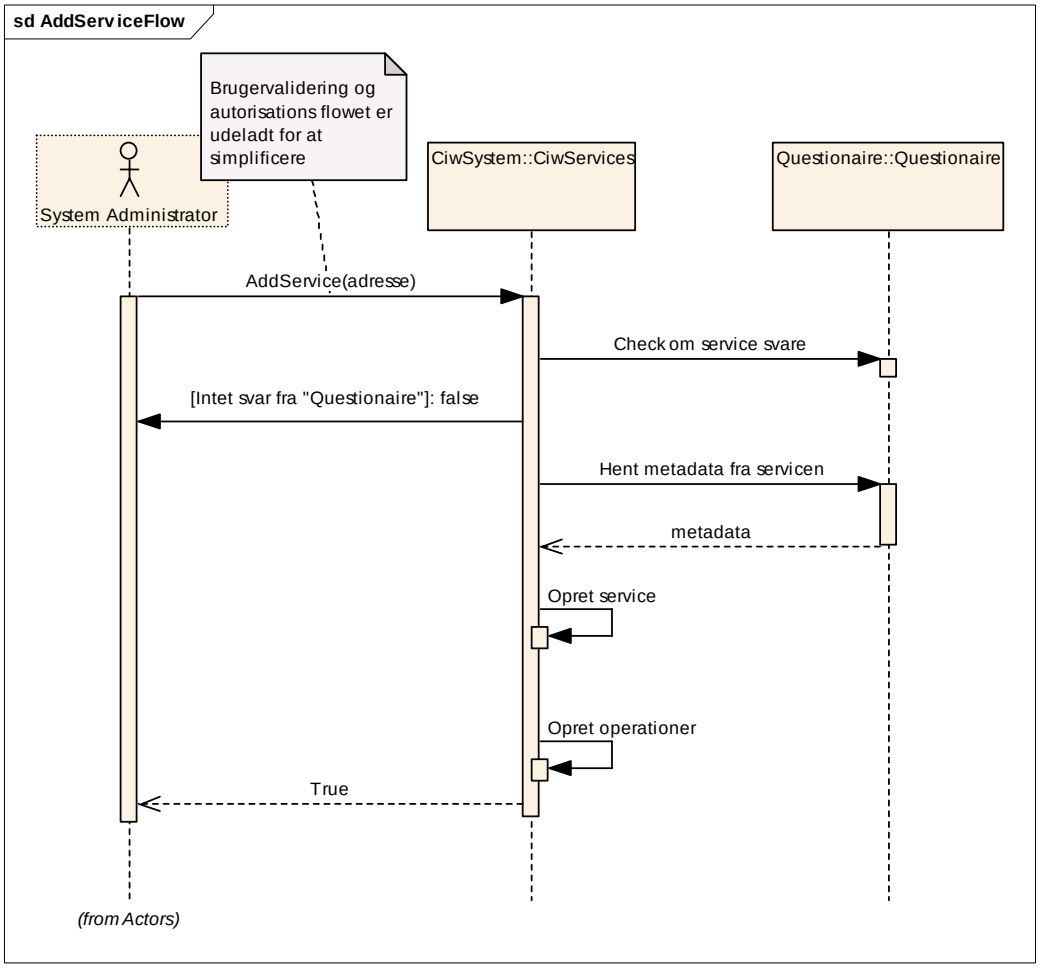


Figur 30 – Klassediagram for systemservicen

Af klassediagrammet ses det, at *CiwSystem* klassen implementerer alle interfaces og dermed kontrakter i systemet. Dette er for at samle alle kontrakter under én service, således der ikke skal opsættes en service til hver kontrakt.

5.3.3 TILFØJ SERVICE TIL SYSTEMET

I Figur 31 ses det hvorledes en service tilføjes systemet. System Administratoren kalder operationen AddService med baseadressen på servicen som parameter. Systemservicen forsøger herefter at indhente information om servicen. Hvis det ikke lykkes at kontakte servicen og indhente information returneres en fejl til system administratoren. Ud fra de indhentede informationer oprettes servicen og dens operationer i systemet.



Figur 31 – System sekvens diagram for tilføjelse af en service til systemet

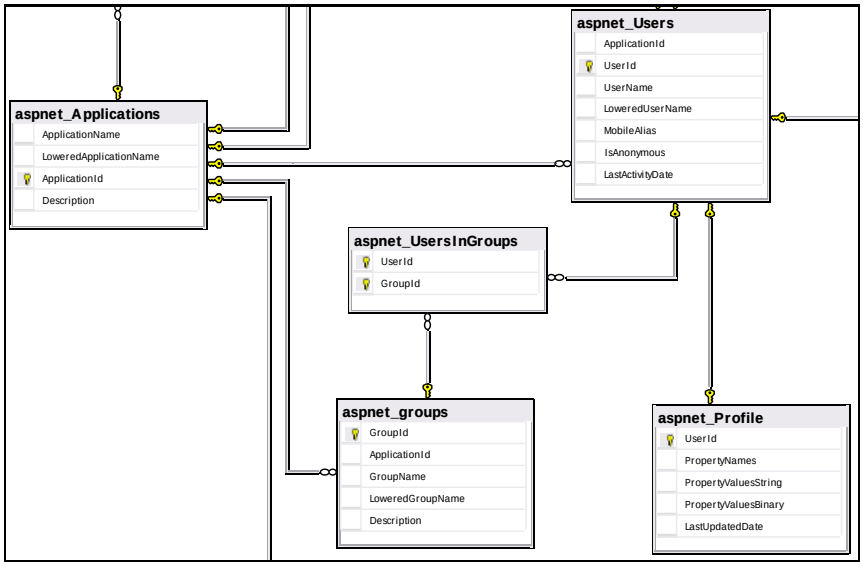
5.3.4 DATABASE DESIGN

Da systemservicen benytter SqlMembership og SqlRoleProvider og disse inkluderer deres eget database design, er der taget udgangspunkt i dette. Udvidelserne, der er foretaget for at understøtte grupper, services og operationer, indebærer en række nye tabeller og tilføjelse af relationer mellem disse og de eksisterende tabeller. Dette har ikke indflydelse på funktionaliteten af SqlMembership og SqlRoleProvider, da der ikke ændres i de tabeller, de benytter.

5.3.4.1 Grupper

Der er foretaget udvidelser for at understøtte grupper i databasen, følgende tabeller er tilføjet:

- **aspnet_groups**
- **aspnet_UsersInGroups**



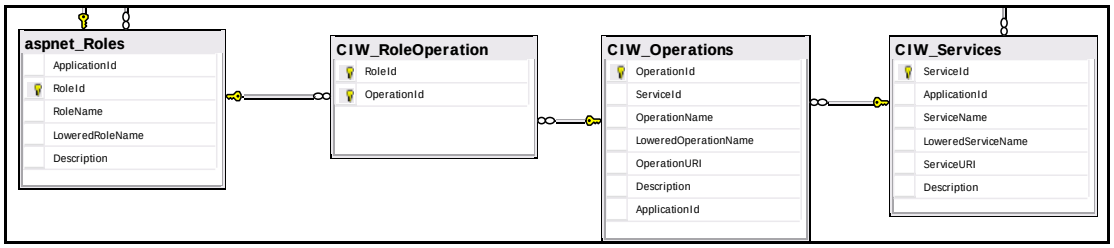
Figur 32 – Gruppe udvidelser for system database

Det ses af Figur 32 og det fulde diagram i bilag 4 afsnit 9.4.1, hvorledes gruppe understøttelser er tilføjet til den eksisterende database.

5.3.4.2 Services og operationer

For at understøtte services, operationer og rettigheder til disse operationer er der i databasen tilføjet følgende tabeller.

- **CIW_Services**
- **CIW_Operations**
- **CIW_RoleOperation**



Figur 33 – Service udvidelser for systemservice

Figur 33 viser de tilføjede tabeller. I bilag 4 afsnit 9.4.1 ses det fulde database diagram. Det ses hvorledes krydstabellen CIW_RoleOperation benyttes til at kæde roller og operationer sammen, således dette kan realisere tildelingen af rettigheder til at udføre operationer.

5.4 QUESTIONAIRESERVICE

Questionaireservicen baserer sig på service skabelonen beskrevet i afsnit 5.2.1. Der vil derfor kun blive behandlet designet af forretningslogikken i servicen.

5.4.1 KONTRAKTER

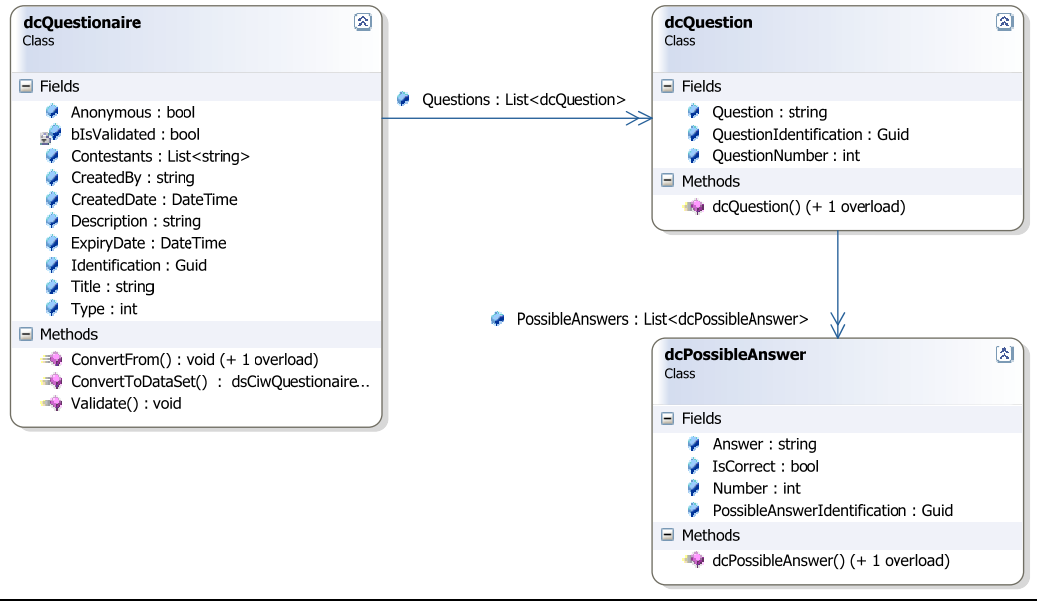
Der er til Questionaireservicen opstillet én servicekontrakt, *IQuestionaire*, som indeholder de operationer servicen skal stille til rådighed. Funktionaliteten implementeres i klassen *Questionaire*. Dette ses i klassediagrammet i Figur 34

5.4.1.1 Datakontrakter

I Questionaireservicen er der designet tre data kontrakter, Datakontrakterne bruges til at udveksle data med servicen. Datakontrakterne er udformet jf. afsnit 4.5.2.1.2

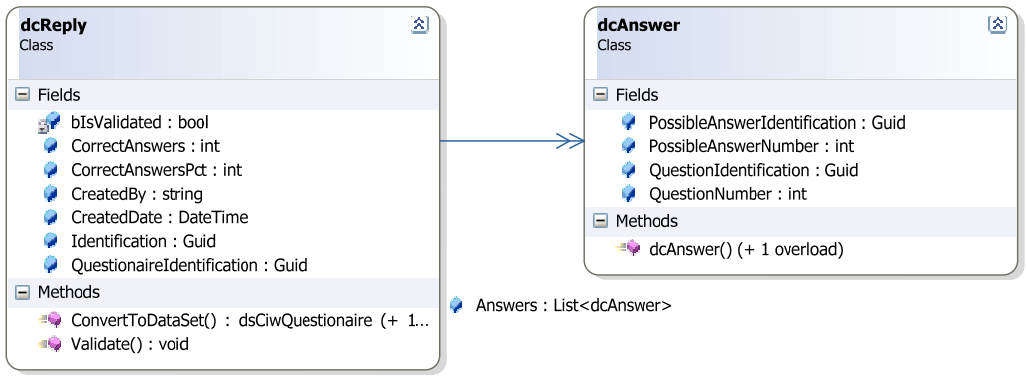
Tabel 7 - Datakontrakter vedrørende test og analyser

dcQuestionaire	dcQuestion	dcPossibleAnswer
dcQuestionaire repræsenterer en test eller analyse og indeholder derfor en liste af spørgsmål i form af typen *dcQuestion*. Klassen indeholder desuden en række hjælpefunktioner til at lette arbejdet med data bla. i forbindelse med database operationer. Disse funktioner har ingen indflydelse på datakontrakten, og benyttes udelukkende internt i servicen.	*dcQuestion* repræsenterer et spørgsmål i en test eller analyse. Til et spørgsmål er der en række mulige svar som er repræsenteret ved en liste af *dcPossibleAnswer*	



Tabel 8 – Datakontrakter vedrørende besvarelse af test og analyser

dcReply	dcAnswer
dcReply benyttes ved besvarelse af en test eller analyse. Den indeholder derfor en række *dcAnswer*	*dcAnswer* er svar på et enkelt spørgsmål i en test eller analyse

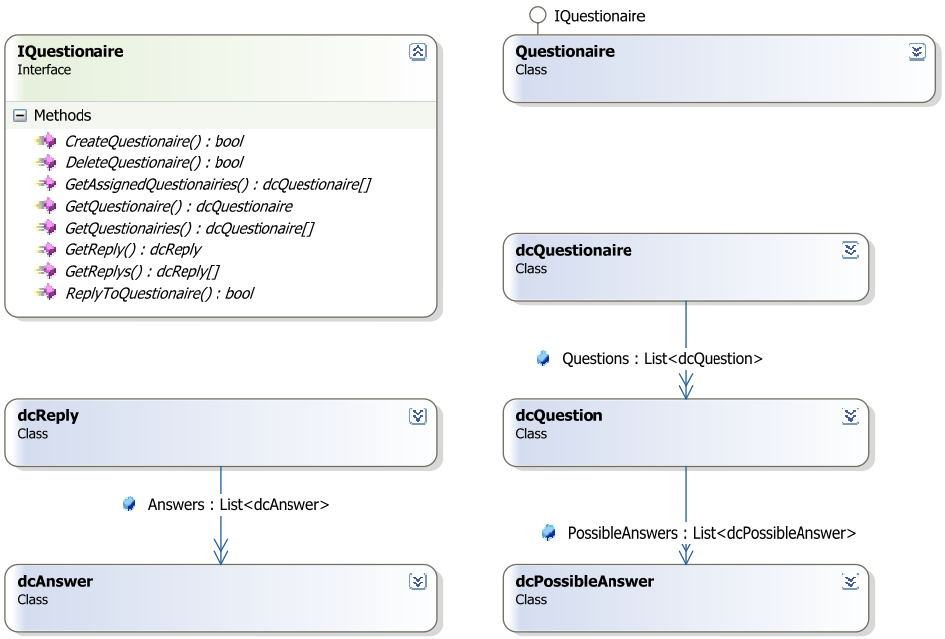


5.4.2 KLASSEDIAGRAM

I tabellen herunder er listet de klasser, der er opstillet til systemservicen, samt en beskrivelse af, hvad formålet er med klassen. Klassediagrammet for systemservicen er vist i Figur 34.

Tabel 9 – Beskrivelse af klasser i Questionaireservicen

Navn	Beskrivelse
IQuestionaire	Datakontrakt med de operationer servicen tilbyder
Questionaire	Implementering af *IQuestionaire* interfacet. Det er i denne klasse forretningslogikken implementeres.



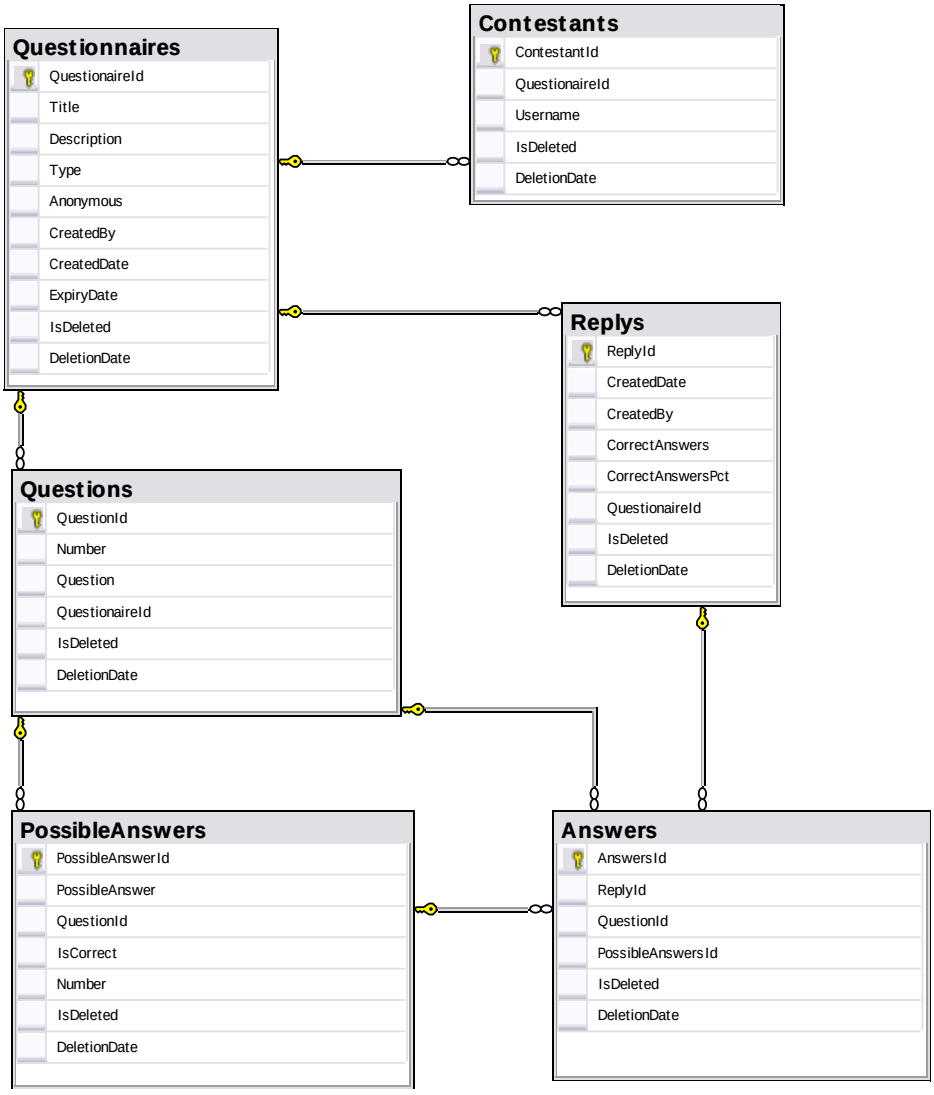
Figur 34 – Klasse diagram for Questionaireservicen

5.4.3 DATABASE DESIGN

På baggrund at domænemodellen fra analysen afsnit 3.4.2 er der foretaget en analyse af hvilke krav, der er til en database, der kan anvendes i servicen. Der er opstillet følgende tabeller, der skal realisere databasen.

- **Questionaires** Egenskaber for en test eller analyse
- **Contestants** Brugere der er tildelt testen eller analysen
- **Questions** Spørgsmål der er stillet i den pågældende test eller analyse
- **PossibleAnswers** Opstillede svarmuligheder til et spørgsmål
- **Replys** Indgivende svar på en test eller analyse
- **Answers** Svar på de stillede spørgsmål

Tabellerne og deres indbyrdes relationer er vist i Figur 35.



Figur 35 – Database diagram for Questionaireservicen

5.5 EVALUERINGSERVICE

Evalueringservicen baserer sig på service skabelonen, beskrevet i afsnit 5.2.1. Der vil derfor kun blive behandlet designet af forretningslogikken i servicen.

5.5.1 KONTRAKTER

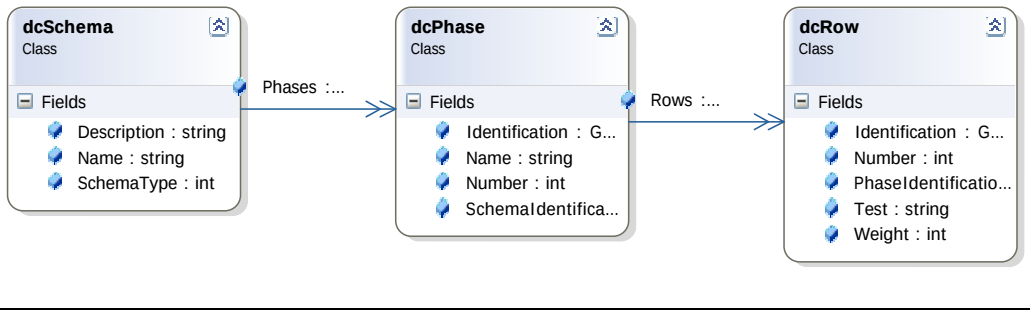
Der er til Evalueringservicen opstillet én servicekontrakt, *IEvaluation*, som indeholder de operationer servicen skal stille til rådighed. Funktionaliteten implementeres i klassen *Evaluation*. Dette ses i klassediagrammet i Figur 36.

5.5.1.1 Datakontrakter

I Evalueringservicen er der designet tre data kontrakter, Datakontrakterne bruges til at udveksle data med servicen. Datakontrakterne er udformet jf. afsnit 4.5.2.1.2

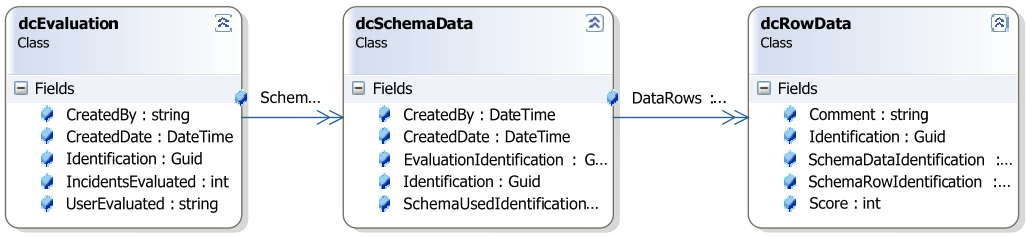
Tabel 10 – Datakontrakter omhandlende administration af skemaer i Evalueringservicen

dcSchema	dcPhase	dcRow
dcSchema beskriver et skema. Et skema indeholder en række faser, og *dcSchema* indeholder derfor en liste af *dcPhase*	*dcPhase* repræsenterer en fase i et skema. En fase har en række punkter; derfor indeholder *dcPhase* en liste af *dcRow*	*dcRow* repræsenterer et punkt i en fase.



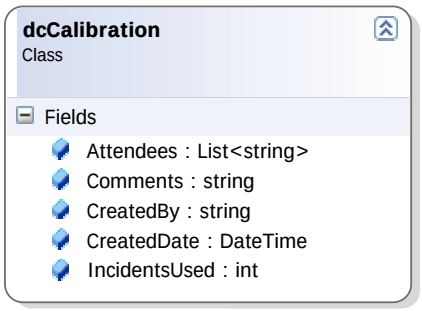
Tabel 11 – Datakontrakter omhandlende evalueringer og skemadata i Evalueringservicen.

dcEvaluation	dcSchemaData	dcRowData
dcEvaluation repræsenterer en evaluering og indeholder bla. en liste af *dcSchemaData*	*dcSchemaData* repræsenterer et udfyldt skema. Indeholder grundlæggende information som hvem, der har udfyldt skemaet, samt hvornår. Til et skema er der en række punkter, som repræsenteres som en liste af *dcRowData*	*dcRowData* repræsenterer data fra et punkt i et udfyldt skema.



Tabel 12 – Datakontrakt til lederkalibrering i Evalueringservicen.

dcCalibration
Indeholder information relevant for en leder kalibrering, bla. en liste af deltagere.

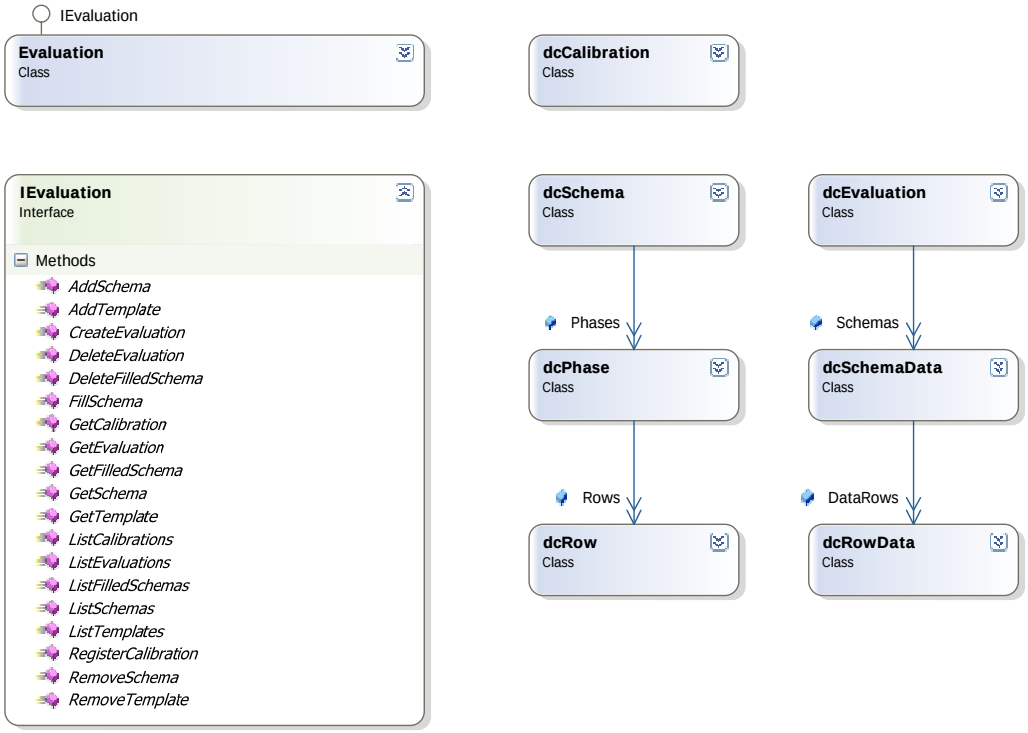


5.5.2 KLASSEDIAGRAM

I tabellen herunder er listet de klasser, der er opstillet til Evalueringservicen, samt en beskrivelse af, hvad formålet er med klassen. Klassediagrammet for systemservicen er vist i Figur 36

Tabel 13 – Beskrivelse af klasserne i Evalueringservicen.

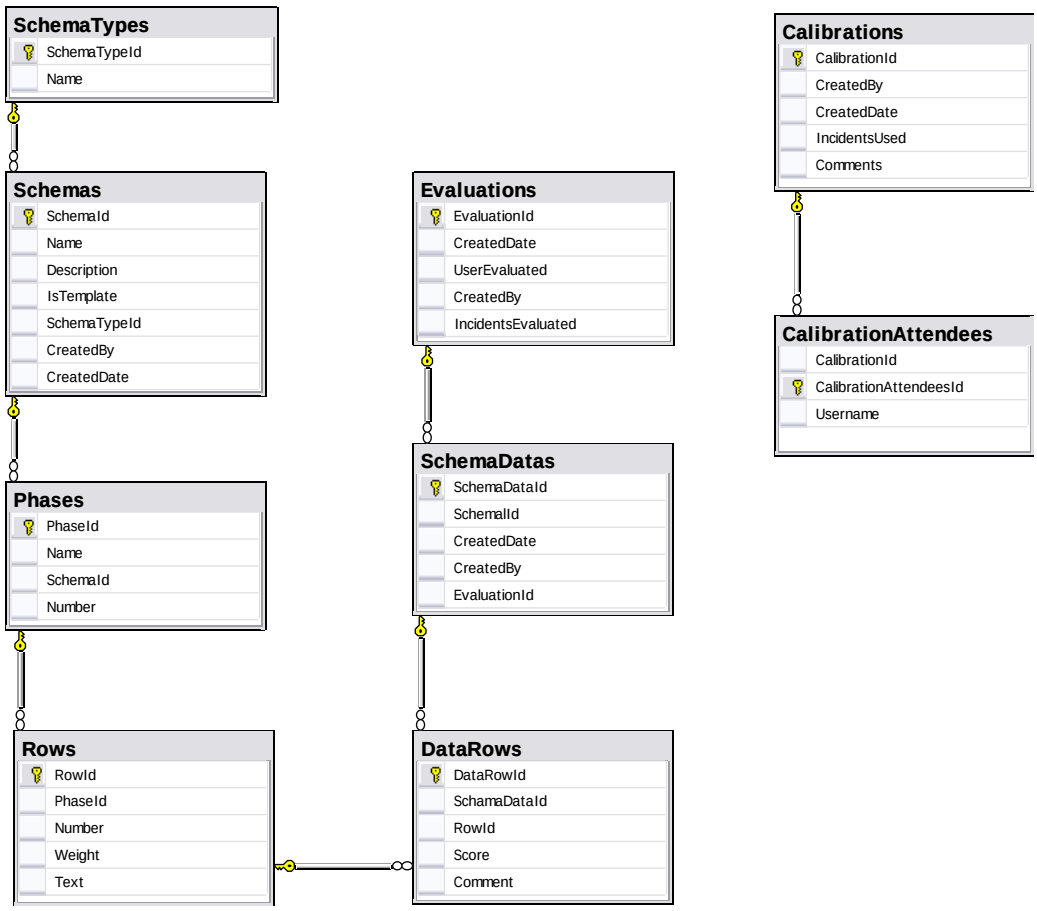
Navn	Beskrivelse
IEvaluation	Servicekontrakt med de operationer servicen indeholder
Evaluation	Implementering af IEvaluation interfacet. Det er i denne klasse forretningslogikken implementeres.



Figur 36 – Klasser i Evalueringservicen

5.5.3 DATABASE DESIGN

På baggrund at domænemodellen fra Figur 18 er der foretaget en analyse af hvilke krav, der er til en database, der kan anvendes i servicen. Der er opstillet følgende tabeller, der skal realisere databasen.



Figur 37 – Database Diagram Evalueringservice

5.6 RESUME

Der er i kapitlet opstillet den overordnede struktur for systemet herunder en sikkerhedsmekanisme, der bygger på, at alle services i systemet foretager validering og autorisering imod systemservicen.

Der er desuden opstillet en skabelon, der kan benyttes ved udvikling af nye services til systemet. Denne skabelon indeholder bla. en assembly til fejlhåndtering og CiwSecurity der muliggør, at benytte den designede sikkerhedsmekanisme i systemet.

Sikkerhedsmekanismen er designet ud fra WCF funktionaliteter, således at den enkelte service ikke skal implementere sikkerhed, men blot konfigureres til at benytte sikkerhedsmekanismen. Dette betyder, at der enkelt kan skiftes til en anden sikkerhedsmekanisme, uden det har indflydelse på implementeringen af servicen.

Systemservicen er designet således, at det er muligt, at tilføje services til systemet. Herefter hentes der automatisk information om denne service, som gemmes i systemet. Således der på baggrund af. Dette, kan tildeles rettigheder til at benytte servicen.

Ud fra ovenstående er det muligt at realisere de stillede krav til systemet, og der er ligeledes med den service orienterede arkitektur opnået den fleksibilitet der er ønsket, således at systemet kan tilpasses med nye services efter behov.

Kapitel 6

IMPLEMENTERING

Der vil i kapitlet være en beskrivelse af hvorledes udvalgte områder af systemet er implementeret. Der vil fokuseres på hvorledes den i designet opstillede arkitektur og sikkerhedsmekanisme er implementeret.

6.1 WCF IMPLEMETERING

6.1.1 ARKITEKTUR

For at opnå den ønskede overordnede arkitektur beskrevet i afsnit 5.2, skal der opsættes en policy for servicerne som beskrevet i afsnit 4.5.2.

Dette gøres i konfigurationsfilerne, hvor der defineres dels hvorledes der kommunikeres med servicen selv, og dels hvorledes den kommunikerer med systemservicen. Desuden opsættes brugervalidering/autorisation, således sikkerhedsmekanismen beskrevet i afsnit 5.2.1 kan benyttes.

Herunder følger eksempler fra Questionaireservicens konfigurations fil. Der viser hvorledes servicen er konfigureret

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

   <appSettings>
     <!--baseaddress for service  -->
     <add key="questbaseAddress" value="http://localhost:667/CiwQuest" />

   </appSettings>

   <!--Database connection string til Questionaire-->
   <connectionStrings>
     <add name="ICEMan"
     connectionString="server=iceman;database=Ciw_Questionaire;integrated security=sspi;"/>
   </connectionStrings>
```

Servicens baseadresse sættes som en *appSetting,* således den kan slås op ved self-hosting. Desuden er der opsat en *connectionstring* til databasen som Questionaire benytter

```
   <system.serviceModel>
   <!--Opsætningen af servicen selv-->
   <services>
      <service name="Ciw.CiwService.Questionaire.Questionaire"
               behaviorConfiguration="QuestServiceBehavior">
        <endpoint address="" binding="wsHttpBinding"
             contract="Ciw.CiwService.Questionaire.IQuestionaire"
             bindingConfiguration="wsHttpBinding_ICiwQuest" />
      </service>
     </services>

   <!--Bindings-->
     <bindings>
       <wsHttpBinding>
         <!--Binding til System servicen Ciw.CiwSystem.CiwSecurity-->
         <binding name="WSHttpBinding_ICiwSecurity">
           <security mode="Message">
             <message clientCredentialType="UserName"/>
           </security>
         </binding>
```

Servicen og dens *endpoint* opsættes, og der angives hvilken *behaviorConfiguration* og bindin,g der skal benyttes. *Address* udfyldes ikke, og på den måde vil *endpointet* blot have servicens baseadresse som adresse. Bindingen er af typen *wsHttpBinding*, der benyttes besked sikkerhed. For at sikkerhedsmekanismen i systemet fungere skal klient *legitimationstypen* være *UserName.*

```
    <!--behavior til Questionaire-->
    <behaviors>
      <behavior name="QuestServiceBehavior" returnUnknownExceptionsAsFaults="true" >
        <serviceCredentials>
          <!--Sæt username/password som authentication, og benyt
          Ciw.Security.CiwUserNamePasswordValidator-->
          <userNameAuthentication userNamePasswordValidationMode="Custom"
          customUserNamePasswordValidatorType="Ciw.Security.CiwUserNamePasswordValidator,
          Ciw.Security" />

        <!--Hent SSL certifikat servicen skal benytte til at authenticate sig overfor klienten-->
        <serviceCertificate findValue="localhost" storeLocation="CurrentUser" storeName="My"
                x509FindType="FindBySubjectName" />
        </serviceCredentials>

        <!-- Sæt Ciw.Security.CiwUserAutorisationManager som autoriserings mekanisme-->
        <serviceAuthorization
        serviceAuthorizationManagerType="Ciw.Security.CiwUserAuthorizationManager, Ciw.Security">
        </serviceAuthorization>
      </behavior>
    </behaviors>
  </system.serviceModel>
</configuration>
```

Behavior delen af konfigurationen angiver hvordan klienter, der benytter Questionaireservicen, skal valideres og autoriseres. Dette skal sættes til at benytte *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* som beskrevet i afsnit 5.2.1.

For at *CiwUserNamePasswordValidator* og *CiwUserAuthorizationManager* kan kommunikere med systemservicen, skal der konfigureres et *endpoint,* som beskriver hvordan der kommunikeres med systemservicen.

```
    <!--Config af forbindelse til System servicen Ciw.CiwSystem.CiwSecurity-->
    <client>
      <endpoint address="http://deploy.nus.local:666/Security" binding="wsHttpBinding"
                bindingConfiguration="WSHttpBinding_ICiwSecurity"
                contract="Ciw.Security.CiwSecurity.ICiwSecurity"
                name="WSHttpBinding_ICiwSecurity">
        <identity>
          <dns value="deploy.nus.local" />
        </identity>
      </endpoint>
    </client>
```

Her konfigureres forbindelsen til systemservicens *endpoint,* som illustreret i Figur 25, således brugervalidering og autorisering kan foretages imod systemservicen.

6.2 VALIDERING/AUTORISERING I SIKKERHEDSMEKANISMEN

For at validere brugernavn og password i sikkerhedsmekanismen anvendes *CiwUserNamePasswordValidator* klassen. Ligeledes benyttes *CiwUserAutorizationManager* klassen til autorisering af brugere.

Disse klasser er som beskrevet i afsnit 5.2.2.1.1, nedarvet fra WCF klasserne *UserNamePassWordValidator* og *ServiceAuthorizationManager*.

6.2.1 *CIWUSERNAMEPASSWORDVALIDATOR*

I den nedarvede *CiwUserNamePasswordValidator* klasse skal funktionen *Validate* override's. I denne funktion implementeres herefter den ønskede funktionalitet til brugervalidering ud fra brugenavn og password.

Eksempel 4 – Implemtering af CiwUserNamePasswordValidator

```
public class CiwUserNamePasswordValidator : UserNamePasswordValidator
{
   public override void Validate(string userName, string password)
   {
      if (null == userName || null == password)
      {
         throw new ArgumentNullException();
      }

      CiwSecurity.CiwSecurityProxy Proxy = new
         Ciw.Security.CiwSecurity.CiwSecurityProxy();

      if (Proxy.AuthenticateUser(userName, password))
      {
      }
      else
      {
         throw new SecurityTokenException("Unknown Username or Password");
      }
   }
}
```

I Eksempel 4 ses det hvorledes *CiwUserNamePasswordValidator* benytter systemservicen via en proxyklasse som beskrevet i afsnit 5.2.2.1.1.

6.2.2 CIWSERVICEAUTHORIZATIONMANAGER

I den nedarvede *CiwUserAutorizationManager* klasse skal funktionen *CheckAccessCore* override's. I denne funktion implementeres herefter den ønskede funktionalitet til autorisation.

Eksempel 5 – Implemtering af CiwUserAuthorizationManager

```
namespace Ciw.Security
{
   public class CiwUserAuthorizationManager : ServiceAuthorizationManager
   {
      protected override bool CheckAccessCore(OperationContext operationContext)
      {
         if (!operationContext.ServiceSecurityContext.PrimaryIdentity.IsAuthenticated)
            return false;

         string sUsername =
            operationContext.ServiceSecurityContext.PrimaryIdentity.Name.ToString();
         string sOperation =
            operationContext.RequestContext.RequestMessage.Headers.Action.ToString();

         CiwSecurity.CiwSecurityProxy Proxy = new
         Ciw.Security.CiwSecurity.CiwSecurityProxy();

         return  Proxy.AuthorizeUser(sUsername, sOperation);
      }
   }
}
```

I Eksempel 5 ses det hvorledes brugernavn og den operation, der ønskes autorisation til at udføre, udtrækkes. Herefter benyttes systemservicen via en proxyklasse, som beskrevet i afsnit 5.2.2.1.1, til at kontrollere, om brugeren må udføre den ønskede operation.

6.3 ADDSERVICE OPERATION

I Eksempel 6 ses implementeringen af *AddService* operationen fra systemservicen. Operationen er implementeret ud fra System Sekvens diagrammet i bilag 8.

Eksempel 6 – AddService operation i systemservicen.

```
[OperationBehavior]
public bool AddService(string sFriendlyName, string sDescription, string sAddress)
{
   dcCiwService newService = new dcCiwService();
   newService.Name = sFriendlyName;
   newService.Description = sDescription;
   newService.Address = sAddress;

   string sMexAddress = sAddress;

   if (sMexAddress.Substring(sMexAddress.Length - 1, 1).Equals("/"))
   {
      sMexAddress += "mex";
   }
   else
   {
      sMexAddress += "/mex";
   }

   //Contract collection
   Collection<ContractDescription> Contracts = new Collection<ContractDescription>();

   try
   {
      // Create a MetadataTransferClient for retrieving metadata.
      EndpointAddress mexAddress = new EndpointAddress(sMexAddress);
      MetadataTransferClient mexProxy = new
               MetadataTransferClient(mexAddress.Uri.Scheme);

      // Retrieve the metadata for all endpoints using metadata exchange protocol
      MetadataSet metadataSet = mexProxy.GetMetadata(new
               MetadataReference(mexAddress));

      //Convert the metadata into contracts
      WsdlImporter importer = new WsdlImporter(metadataSet.MetadataSections);
      Contracts = importer.ImportAllContracts();
   }
   catch (System.UriFormatException expURI)
   {
      CiwSystemExceptionHandler.HandleException(expURI,"CiwService,
               AddService",false);
   }
   catch (InvalidOperationException expInvOp)
   {
      CiwSystemExceptionHandler.HandleException(expInvOp,"CiwService,
               AddService",false);
   }
   catch (Exception e)
   {
      CiwSystemExceptionHandler.HandleException(e,"CiwService, AddService",true);
   }
```

Der forsøges at oprette forbindelser til metadataexchange endpointet på den angivne adresse. Adressen skal angives i en *dcCiwService* datakontrakt.

Kan der oprettes forbindelse, benyttes der en *MetadataTransferClient* til at hente metadata information om servicen på adressen. Herefter benyttes en *WsdlImporter* til at udtrække alle kontrakter fra metadataene.

```
//List to hold all operations across all contracts
List<OperationDescription> allOperations = new List<OperationDescription>();

//Iterate thru the contracts and add operations to the list
foreach (ContractDescription CD in Contracts)
{
   foreach (OperationDescription OD in CD.Operations)
   {
      allOperations.Add(OD);
   }
}
```

For hver kontrakt udtages alle *OperationDescription* objekterne og de lægges i en samlet liste kaldet *allOperations*.

```
newService.Operations = new List<dcCiwOperation>();

//Add the operations from the list to the newService object
for (int i = 0; i < allOperations.Count; i++)
{
   dcCiwOperation newOperation = new dcCiwOperation();
   newOperation.ServiceName = newService.Name;
   newOperation.Name = allOperations[i].Name;
   newOperation.Action = allOperations[i].Messages[0].Action;
   newOperation.Description = "Insert description here.";
   newService.Operations.Add(newOperation);

}
```

Operationerne i *allOperations* tilføjes til *newService* objektet, med information om den enkelte operation indsat.

```
//Insert service in database
try
{
   dsCiwServicesTableAdapters.CIW_ServicesTableAdapter TA = new
   Ciw.CiwService.CiwSystem.dsCiwServicesTableAdapters.CIW_ServicesTableAdapter();

   Guid? newGuid = new Guid();

   TA.Insert("/Membership2", newService.Name, newService.Address,
            newService.Description, ref newGuid);

   //insert operations in database
   AddOperations(newService);

   return true;
}
catch (System.Data.SqlClient.SqlException e)
{
   CiwSystemExceptionHandler.HandleException(e, "CiwService, AddService", true);
}
catch (Exception e)
{
   CiwSystemExceptionHandler.HandleException(e, "CiwService, AddService", true);
}
```

Servicen indsættes i databasen og vha. *AddOperations* metoden indsættes de tilhørende operationer i databasen.

6.4 DATA ACCESS

De fleste services der skal indgå i systemet har behov for et datastore. Der anvendes i den forbindelse Microsoft SQL Server 2005. Hver service har egen database og skal også selv implementere dataaccess funktionalitet.

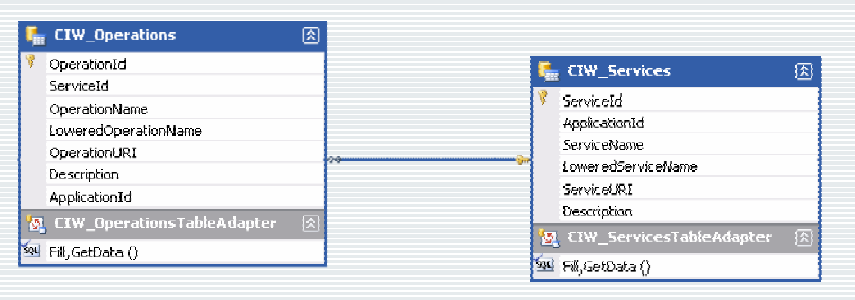
6.4.1 STORED PROCEDURES

Datamanipulering foregår gennem Stored Procedures. Der er til dataudveksling med databasen implementeret stored procedures.

En del af de stored procedures er genereret ved hjælp af OlyMars[6], der er et værktøj som kan generere stored procedures ud fra tabellerne i en database.

6.4.2 DATASET

Der arbejdes i servicerne med .NET *Dataset* til at manipulere med data i databaserne. Når der arbejdes med dataset i Visual Studio defineres den relevante databasestruktur i XSD (XML schema) format, dette kan gøres visuelt direkte i Visual Studio som vist i Figur 38 hvor et dataset fra systemservicen kan ses.



Figur 38 – Dataset til CiwServices klassen

På hver tabel i datasettet er der mulighed for at angive hvilket stored procedures der skal benyttes til insert, update, delete og select operationer.

6.5 RESUME

I kapitlet er der gennemgået hvorledes implementeringer er udført for at opnå det i kapitel 5 opstillede design. Specifikt er der gennemgået hvorledes arkitekturen er realiseret, og hvorledes sikkerhedsmekanismen defineret i designet er implementeret.

[6] http://www.olymars.net

Kapitel 7

TEST

Der vil i kapitlet blive beskrevet hvilke typer test af systemet der er planlagt. Da systemet endnu ikke har nået et stadie, hvor det er relevant at udføre komplette tests af systemet, er der i kapitlet opstillet hvorledes de forskellige typer test bør udføres. Dette kan således påbegyndes, når systemet når et stadie, hvor dette er relevant.

7.1 TESTMETODER

Der er planlagt 3 typer test af systemet. Unittest, arkitekturtest og accepttest.

Unittestene er brugt løbende i implementeringen til at verificere de enkelte funktioner i systemets klasser. Der vil i dette kapitel blive beskrevet, hvorledes unittest er blevet benyttet under udviklingen og hvorledes de er kodet.

Arkitekturtesten udføres for at påvise, hvorvidt den overordnede arkitektur og sikkerhedsmekaniske, opstillet i kapitel 5, gør det muligt at benytte denne struktur til et system bestående af services, der kan opfylde de i kapitel 2 opstillede krav.

Accepttesten udføres for at bekræfte om systemets servicer realiserer de krav, der er opstillet i kravspecifikationen i kapitel 2. Der er i projektperioden ikke foretaget accepttest af systemet, da det endnu ikke er på et stadie, hvor dette ville være relevant. I stedet er der beskrevet hvorledes det er tænkt, accepttesten skal udføres

. Unittestene og arkitekturtesten vil blive foretaget med unittest værktøjet i Visual Studio 2005. Accepttesten skal udføres manuelt vha. mockupmodeller af brugerfladen

7.1.1 ARKITEKTURTEST

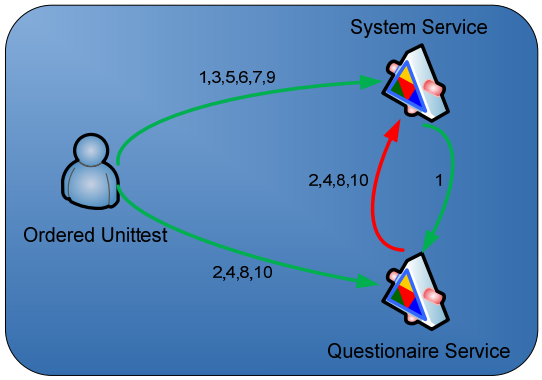
Arkitekturtesten er udformet som en *orderedtest*. En *orderedtest* er en række almindelige unittests, der udføres automatisk af Visual Studio i en valgt rækkefølge. Unittests vil blive beskrevet yderligere i afsnit 7.1.2

Arkitekturtesten udfører en række operationer i systemet for at underbygge den opstillede arkitektur. De unittests, der benyttes i arkitekturtesten, adskiller sig fra de øvrige unittests, ved at de tester imod de hostede services og ikke direkte imod klasserne.

Tabel 14 – Unittests i arkitekturtesten.

#	Unittest navn	Service der kaldes	Formål	Forventet resultat
1	AddServiceServiceTest	System	Tilføjer Questionaireservicen til systemet	Servicen og dens operationer er tilføjet systemet.
2	GetQuestionairiesService-TestUserDontExist	Questionaire	Forsøger at benytte en operation i Questionaire.	Ikke muligt, da brugeren ikke er oprettet med brugernavn/password i systemet og derfor ikke kan valideres.
3	CreateUserServiceTest	System	Opretter brugeren i systemet.	Brugeren er oprettet med brugernavn/password i systemet, således han nu kan valideres.
4	GetQuestionairiesService-TestNoRights	Questionaire	Forsøger igen at benytte operationen i Questionaie, som den bruger der blev oprettet i nr 3.	Ikke muligt. Brugeren kan valideres, men brugere er ikke tildelt nogen roller, og har derfor ikke rettigheder til at benytte operationen i Questionaire.
5	CreateRoleServiceTest	System	Opretter rolle i systemet.	Rollen er oprettet i systemet, således den nu kan tildeles rettigheder.
6	AssigRightsToRoleServiceTest	System	Tildeler rettigheder til den i nr. 5 oprettede rolle, således rollen har rettighed til at kalde operationen i Questionaireservicen.	Rettigheder er tildelt rollen, således brugere med denne rolle nu kan benytte operationen i Questionaire.
7	AddUsersToRolesService Test	System	Tilføj den i nr. 3 oprettede bruger til rollen oprettet i nr. 5.	Brugeren er tildelt rollen.
8	GetQuestionairiesService-Test	Questionaire	Forsøg igen at benytte operationen i Questionaire. Det forventes at dette lykkes, da brugeren nu er tildelt rollen oprettet i nr. 5	Operationen kan benyttes da brugeren nu er tildelt en rolle med rettigheder, således brugeren af systemet kan autoriseres til at benytte operationen.
9	RemoveUsersFromRoles-ServiceTest	System	Fjern rollen fra brugere som der blev tildelt i nr. 7	Brugeren er fjernet fra rollen.
10	GetQuestionairiesService-TestNoRights	Questionaire	Forsøg igen at benytte operationen.	Der er ikke muligt at benytte operationen, da brugeren er fjernet fra den rolle der gav rettigheder til at benytte den.

Tabel 14 viser rækkefølgen af unittests i arkitektur testen, og hver enkelt unittests formål og forventet resultat.



Figur 39 – Illustration af kald under arkitekturtesten

I Figur 39 ses det hvorledes der foretages kald når hver unittest eksekveres. Tallene angiver hvilket nr. unittest der udfører kaldet. Den røde pil angiver det kald, der har med brugervalidering/autorisering at gøre.

7.1.1.1 Udførsel

For at gennemføre arkitekturtesten, skal system og Questionaireservicerne startes. Dette gøres vha. to konsol applikationer som hoster hver deres service. Her efter startes testen i Visual Studio.

7.1.1.2 Resultat.

Testen kan gennemføres uden fejl. Testen er ikke udtømmende, men påviser at nøgle punkter som tilføjelse af servicer til systemet, oprettelse af bruger, roller, rettigheder og bruger validering/autorisering fungerer som ønsket.

7.1.2 UNITTESTS

Unittest er benyttet til at udføre tests direkte af klasserne der implementerer servicerne. Disse test anskueliggør derfor ikke, hvorvidt servicerne fungerer korrekt, når de er sat op og hostet, men kun hvorvidt de enkelte operationer returnerer det forventede output ud fra det input de får. En funktion som *AddService,* der opretter en service i systemet ved at indsætte en række i databasen, vil ikke direkte blive testet for, hvorvidt de ønskede data bliver indsat. Der vil derimod blive testet for, om der bliver returneret de korrekte værdier, og lige så vigtigt, kastet de forventede exceptions ved forkerte input.

Af denne årsag vil unittest ikke blive brugt som accepttest af systemet, men som en række automatiske tests, der hurtigt kan udføres og give et overblik over om der f.eks. er introduceret fejl i systemet efter ændringer i implementeringen. Unittestene vil dog danne grundlag for accepttestene i den forstand, at der ikke vil blive foretaget accepttest før unittests er gennemført succesfuldt.

Unittestene i Visual Studio giver mulighed for at benytte *Code coverage*. Code coverage opsamler under testkørsler information om hvilke linier kode, der er blevet eksekveret og testet. Dette giver udvikleren/testeren mulighed for visuelt at se hvilket linier kode der er testet. Ud over den visuelle hjælp udregnes der af code coverage også hvor stor en procentdel af koden, der er dækket af test.

7.1.2.1 *Implementering af unittests*

Hver funktion i systemet vil blive testet med en eller flere unittestmetoder. Antallet af testmetoder vil afhænge af kompleksiteten af funktionen der testes. En funktion med mange forgreninger i f.eks. if-sætninger eller try/catch blokke vil derfor typisk kræve flere testmetoder, da alle forgreninger skal testes.

Eksempel 7 – Testmetode til AddService funktionen

```
[TestMethod()]
public void AddServiceTest()
{
    CiwServices target = new CiwServices();

    string sFriendlyName = "UnittestAddedservice";
    string sDescription = "Service Added by unittest";
    string sAddress = "http://localhost:667/CiwQuest/";

    bool expected = true;
    bool actual;

    actual = target.AddService(sFriendlyName, sDescription, sAddress);
    Assert.AreEqual<bool>(expected, actual, "Ciw.CiwService.CiwSystem.CiwServices.AddService did
    not return the expected value");
}
```

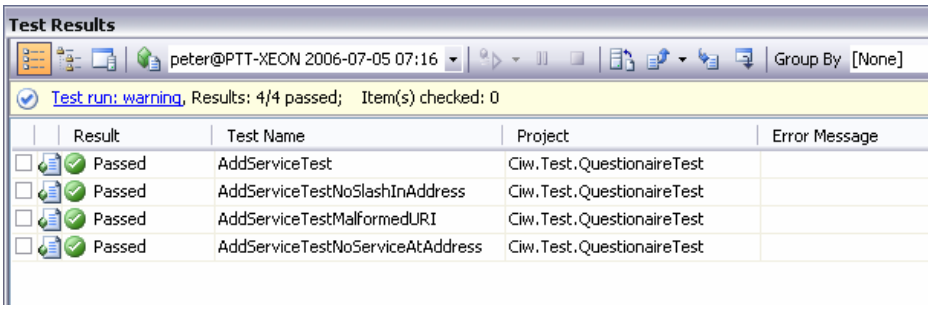
I Eksempel 7 ses koden til en af testmetoderne til *AddService* funktionen. Paramenterne til *AddService* funktionen og forventet output opsættes. Efter *AddService* funktionen er kaldt, evalueres der vha. et *Assert.AreEqual* kald, hvorvidt den returnerede værdi er lig den forventede værdi. Hvis de ikke er lig, fejler testen og fejlbeskrivelsen, der indsættes som parameter til *Assert.AreEqual* udskrives i Visual Studio.

Eksempel 8 – Testmetode til AddService der forventer, der kastes en exception.

```
[TestMethod()]
[ExpectedException(typeof(FaultException<Ciw.CiwService.CiwSystem.CiwServicesException>), "No
service at that address, should throw Endpoint exception")]
public void AddServiceTestNoServiceAtAddress()
{
    CiwServices target = new CiwServices();

    string sFriendlyName = "UnittestAddedservice3";
    string sDescription = "Service Added by unittest";
    string sAddress = "http://imnotaserver:667/CiwQuest/";

    target.AddService(sFriendlyName, sDescription, sAddress);
}
```

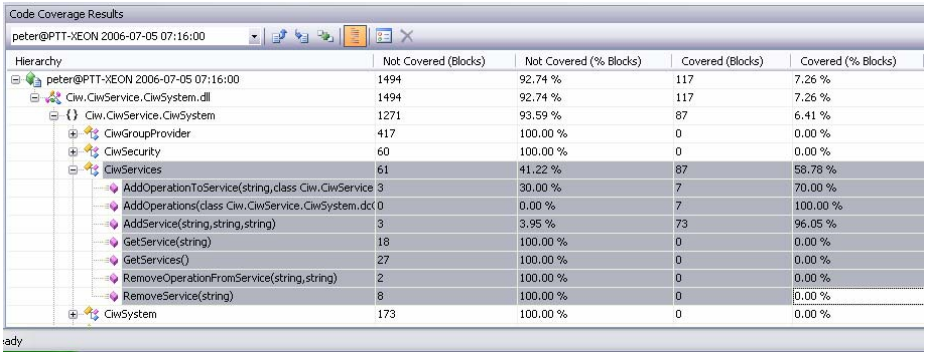
I Eksempel 7 ses koden til en af testmetoderne til *AddService* funktionen. Paramenterne til *AddService* funktionen og forventet output opsættes. Efter *AddService* funktionen er kaldt, evalueres der vha. et *Assert.AreEqual* kald, hvorvidt den returnerede værdi er lig den forventede værdi. Hvis de ikke er lig, fejler testen og fejlbeskrivelsen, der indsættes som parameter til *Assert.AreEqual* udskrives i Visual Studio.

Eksempel 8 viser koden til en af de testmetoder til *AddService,* der forventer at der bliver kastet en exception. Der benyttes ikke *Assert* i denne variant, men i stedet dekoreres metoden med *[ExpectedException]* hvor typen af exception der forventes kastet, og fejlbeskrivelse specificeres.



Figur 40 – Resultat af unittests for AddService

I Figur 40 ses testresultatet fra kørslen af de 4 funktioner der tester *AddService*, og i Figur 41 ses code coverage resultatet.



Figur 41 – Code Coverage resultat efter kørsel af unittests for AddService

Unittests til de øvrige funktioner og klasser i systemet implementeres og udføres som beskrevet her. Det er målet at opnå en Code Coverage på 80-90% inden accepttest af systemet påbegyndes.

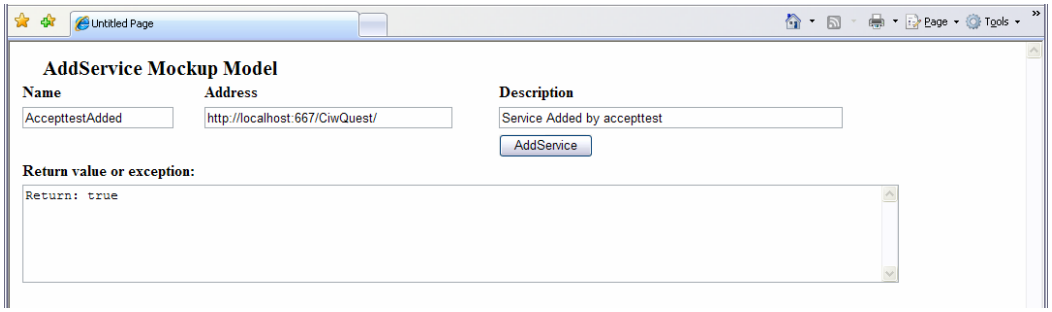
7.1.3 ACCEPTTEST

Accepttesten har til formål at sikre, at de i kravspecifikationen opstillede krav til funktionalitet opfyldes. Accepttesten udføres manuelt ved brug at mockupmodeller af brugerfladen. Desuden vil Microsoft SQL Server Management Studio blive brugt hvor test kræver at data i databasen sammenligner input/output data.

For hvert punkt i kravspecifikationen skal der udføres en accepttest, således det kan bekræftes at alle krav til systemet er opfyldt.

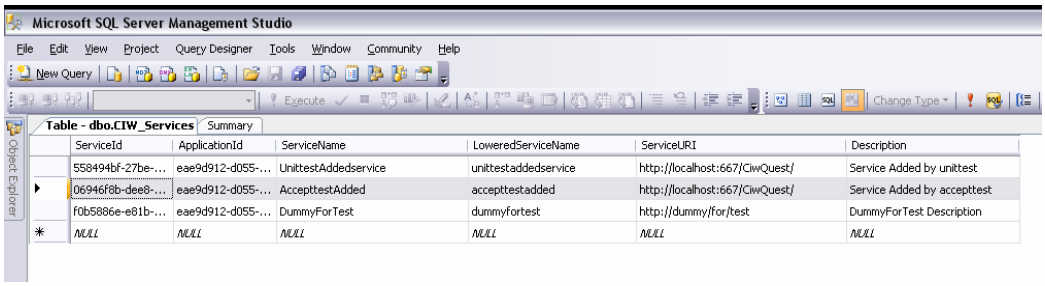
7.1.3.1 Eksempel på udførsel

For at teste f.eks. kravspecifikation punkt RQ-S21 kaldes operationen *AddService* i systemservicen. Systemservicen og servicen der ønskes tilføjet startes og vha. en mockup model kaldes operationen med de krævede parametre. I Figur 42 ses et simpelt eksempel på en mockupmodel der kunne benyttes til testen.



Figur 42 – Mockup model der benyttes til AddService test

Når operationen er udført, sammenholdes inputdata med de data, der er blevet indsat i databasen.



Figur 43 – Manuel kontrol af data i databasen vha. Microsoft SQL Server Management Studio

Figur 43 viser i Microsoft SQL Server Management Studio hvilke data, der er i system databasen i CIW_Services tabellen efter operationen er udført. Ved test af *AddService* operationen skal det i tabellen CIW_Operations ligeledes bekræftes, at de operationer, som servicen tilbyder, er tilføjet korrekt til tabellen.

7.2 RESUME

Der er i kapitlet opstillet hvilket typer test der skal udføres på systemet, samt eksempler på udførelsen. Desuden er der udført en succesfuld arkitekturtest, og ud fra den kan konkluderes, at det er lykkes at realisere den ønskede overordnede struktur.

Kapitel 8

KONKLUSION

8.1 PROBLEMFORMULERING

Projektet har taget udgangspunkt i et oplæg fra Tommy Langhoff omkring udvikling af et system til kvalitetssikring i kundekontaktcentre. Tommy Langhoff har på baggrund af sine erfaringer i kundekontaktcenter branchen udarbejdet oplægget til systemets funktionalitet. Oplægget opstiller en række arbejdsprocesser, der anvendes i kundekontaktcentre til at udvikle den enkelte medarbejders kompetenceniveau. Til hver af disse processer skal der udvikles et værktøj til at styre og udføre den enkelte proces. Alle disse værktøjer danner til sammen det system, som Tommy Langhoff gennem arbejdet med kundekontaktcentre har set behovet for.

Efter gennemgang af materialet, blev der ud fra systemets karakteristika evalueret på mulige løsningsmetoder. Ud fra de relativt individuelle værktøjsbeskrivelser virkede det nærliggende at designe systemet ud fra en Service Orienteret tankegang, således at værktøjerne kunne realiseres som individuelle services. På baggrund af disse antagelser har projektet taget udgangspunkt i at undersøge, om man med fordel kan anvende en service orienteret arkitektur, udviklet med Windows Communication Foundation, til et system som dette.

8.2 OPSUMMERING

På baggrund af de i kapitel 1 identificerede services, blev der opstillet kravspecifikation til de enkelte services ud fra materialet fra Tommy Langhoff. Denne kravspecifikation dannede herefter grundlag for den videre udvikling af systemet.

I analysen blev der opstillet aktør definitioner, domæne modeller og use cases ud fra grundlæggende principper for objekt orienteret analyse.

Aktør definitionerne blev opstillet på tværs af alle services, da det er antaget, at servicerne udgør et samlet system, og de således er rettet mod den samme gruppe aktører. Derimod blev analysen af de enkelte services foretaget separat ud fra antagelsen om, at en service kunne betragtes som et isoleret domæne der løser et forretningsspecifikt behov. Disse services har en række ikke forretningsspecifikke behov, som løses af den opstillede systemservice.

Efter analysen kan det konkluderes, at begge antagelser var korrekte, da der under analysen af servicerne ikke opstod behov for yderligere aktører end dem, der allerede var opstillet for det samlede system. Ligeledes tegnede der sig ikke yderligere ikke funktionelle behov i servicerne, end dem der allerede var defineret som systemservicen.

Da service orienteret arkitektur og Windows Communication Foundation var nyt for os begge, blev, der inden designet blev påbegyndt, indsamlet viden og praktisk erfaring om netop SOA og WCF. Dette skete i form af bøger og foredrag, men da WCF var, og stadig er i beta, har mængden af trykt litteratur været begrænset, hvorfor en stor del har måttet findes som artikler på Internettet.

Der blev desuden foretaget en lang række forsøgsimplementeringer med WCF for at få kendskab til teknologiens muligheder og begrænsninger, inden designet af systemet blev påbegyndt.

Ud fra analysen, den indsamlede viden og praktisk erfaring med SOA og WCF, blev designet opstillet. Designet af forretningslogikken i de enkelte services blev foretaget ud fra objekt orienteret design principper, mens der for den overordnede arkitektur blev fokuseret på Service Orientering. Den overordnede arkitektur blev opstillet og servicekontrakter, datakontrakter og klassediagrammer udarbejdet ud fra analysen.

Under udarbejdelsen af designet blev det klart, at det var en stor fordel, at der var blevet benyttet en del tid på at fordybe sig i WCF, inden designet blev på begyndt. Dette betød, at der

på baggrund af forsøgsimplementeringerne og de erfaringer, der var gjort, var muligt at udarbejde designet således, at de problemer, der var opstået under forsøgsimplementeringerne, kunne undgås i den endelige implementering.

Der blev opstillet en serviceskabelon til at lette udviklingen af services til systemet. Denne indeholder fejlhåndtering og sikkerhed.

Implementeringen af systemservicen var i store træk at samle kode fra en række af forsøgsimplementeringerne, da mange af aspekterne i designet og implementeringen af systemservicen var dækket af forsøgsimplementeringerne.

Implementeringen af Questionarie og Evalueringservicerne tog udgangspunkt, i den i designet opstillede serviceskabelon. Implementeringen af den specifikke forretningslogik startede fra bunden med udgangspunkt i designet.

Der er ikke foretaget en komplet test af systemet, men opstillet metoder for hvorledes systemet kan testes. Der er gennemført en arkitekturtest der efterviser problemstillingen. Denne test fokuserer på vigtige punkter i arkitekturen som brugervalidering og autorisering.

8.3 KONKLUSION

Det er lykkedes at skabe et fundament til et system, som tager udgangspunkt i de fundamentale aspekter i service orienteret arkitektur. Vi har i projektet designet systemet således, at de funktionelle krav til systemet er opfyldt, men også følger de generelle principper i service orientering, så det er muligt at drage nytte af de fordele dette giver.

Der er i overensstemmelse med de opstillede krav og ønsker til løsningen blevet designet et system, der består af services med hver deres formål. Der er blevet brugt mest tid på system-servicen, da denne service er fundamental for systemet. Der er i designet og implementeringen lagt vægt på at benytte de funktionaliteter WCF og .NET stiller til rådighed, frem for selv at implementere lignende funktionalitet. Eksempelvis .NET's MembershipProvider, da denne tilbyder funktionalitet der direkte kan benyttes i systemet.

Systemet er tilstrækkeligt implementeret til, at eftervise problemstillingen. På baggrund af det succesfulde resultat af arkitekturtesten, er det påvist, at den opstillede service orienterede arkitektur kan benyttes til realisering af kravene til Evaluering og Questionaire. Da de resterende services i den samlede kundekontaktcenter løsning, kun adskiller sig fra Questionaire og Evaluering i deres forretningslogik, kan disse implementeres efter samme principper. På baggrund af dette kan det konkluderes, at der kan anvendes en service orienteret arkitektur implementeret med WCF til at udvikle et system til kundekontaktcentre med ud fra Tommy Langhoffs visioner.

Vi ser resultatet af projektet, som en verificering af, at det ved hjælp af Windows Communication Foundation og service orientering er muligt, at opbygge et system som kan realisere de krav der er stillet i oplægget. Løsningen vil desuden muliggøre, at man på sigt kan udvide systemet uden, at det vil have indflydelse på de allerede kørende services. Anvendelsen af WCF og service orientering har ud fra et implementerings synspunkt øget mulighederne for vedligehold og videreudvikling betragteligt.

Ud fra de erfaringer, vi har opnået gennem projekt perioden, er det vores overbevisning, at service orientering har tilført løsningen mere end blot at realisere de fastsatte krav. Systemet har desuden opnået en fleksibilitet der gør det muligt, at tilpasse systemet forskellige behov, fra kunde til kunde.

Service orienteringen gør desuden genbrug af de enkelte services i en anden kontekst oplagt. Systemservicen vil ligeledes nemt kunne bruges som fundament til andre løsninger, da denne er gjort fuldstændig domæne uafhængig.

8.4 FREMTIDIGT ARBEJDE

- **Fejlhåndtering og logning**
  Ved projektets afslutning, er der implementeret et minimum af fejlhåndtering og logning. Der skal i den videre udvikling af systemet opstilles en model og retningslinier for, hvorledes dette skal håndteres i systemet. Denne model skal ligge til grund for implementeringen af fejlhåndtering og logning i systemets services.

- **Implementering af resterende services**
  De i projektet behandlede services skal færdig implementeres, således de opfylder alle stillede krav til dem. For de resterende services identificeret i afsnit 1.6, skal der foretages analyse, design og implementering efter samme principper, som de i projektet behandlede services.

- **Udvikling af brugerflade**
  I samråd med Tommy Langhoff og ud fra modeller opstillet af ham, skal der designes og udvikles brugerflade til systemet.

- **Pilotkunder skal teste systemet**
  Under den videre udvikling af systemet, vil en række pilotkunder udfører test af systemet, således fejl og uhensigtsmæssigheder i systemet kan afklares inden den endelige lancering af systemet.

- **Videreudvikling af systemservicen**
  Systemservicen skal videreudvikles med bla. yderligere administrative funktionaliteter, som f.eks. opstilling af faktureringsgrundlag, således der ud fra antallet af brugere kunden har på systemet, kan foretages fakturering.

KILDEFORTEGNELSE

[1] David Pallmann
"Programming 'Indigo'"
Microsoft Press, 2005

[2] Craig Larman
"Applying UML and patterns"
Prentice Hall, 2002

[3] Martin L. Shoemaker
"UML Applied – A .NET Perspective"
Apress, 2004

[4] Joachim Rossberg and Rickard Redler
"Pro scalable .NET 2.0 application designs"
Apress, 2006

[5] Martin Fowler
"Patterns of enterprise application architecture"
Addison-Wesley, 2003

[6] David Chapelle
"Introducing Indigo: An early look"
http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnlong/html/introindigov1-0.asp

[7] Microsoft Corporation
"Introduction to the Windows Communication Foundation"
http://wcf.netfx3.com/files/folders/product_team/entry1864.aspx

[8] Microsoft Corporation
"Programming the Windows Communication Foundation"
http://wcf.netfx3.com/content/resources.aspx

[9] Clemens Vasters
"Introduction to building Windows Communicatio Foundation services"
http://msdn.microsoft.com/library/en-us/dnlong/html/introtowcf.asp

[10] John Evdemon
"The Four Tenets of Service Orientation"
http://www.bpminstitute.org/articles/article/article/the-four-tenets-of-service-orientation.html

[11] Microsoft Corporation
"Windows Communication Foundation"

http://windowssdk.msdn.microsoft.com/en-us/library/ms735119.aspx

[12] Laurence Moroney
"The Windows Communication Foundation: A Primer"
http://www.devx.com/dotnet/Article/29414

[13] Easwaran G. Nadhan
"Service-Oriented Architecture: Implementation Challenges"
http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj2soaimpc.asp

[14] David S. Linthicum
"8 things most people misunderstand about SOAs"
http://www.soainstitute.org/articles/article/article/8-things-most-people-misunderstand-about-soas.html

[15] David Sprott and Lawrence Wilkes
"Understanding Service-Oriented Architecture"
http://msdn.microsoft.com/architecture/soa/default.aspx?pull=/library/en-us/dnmaj/html/aj1soa.asp

[16] W3C
"Web Services Description Language (WSDL) 1.1"
http://www.w3.org/TR/wsdl

[17] IBM Corporation
"Patterns: Service-Oriented Architecture and Web Services"
http://www.redbooks.ibm.com/redbooks/SG246303/wwhelp/wwhimpl/java/html/wwhelp.htm

[18] Microsoft Corporation
"Web Service Specifications"
http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx

[19] Thomas Erl
"Second generation (WS-*) specifications"
http://www.soaspecs.com/

[20] Achim Oellers
Slideshows ved Microsoft seminar "Sikker og pålidelig SOA i hverdagen" 5 April 2006.

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/Contract_Policy_Addressing.ppt

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Autonomy.ppt

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Edges.PPT

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Exceptions_and_Distributed_Fault_Management.PPT

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Intro.ppt

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Security.ppt

http://download.microsoft.com/download/2/f/4/2f4b5563-10bf-47fc-9355-1b3f1c56c802/SOA2_Service_Patterns.PPT

[21] W3C
”SOAP Version 1.2 Part 1: Messaging Framework”
http://www.w3.org/TR/soap12-part1/

[22] W3C
“UDDI Specifications”
http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

Kapitel 9

BILAG

9.1 BILAG 1 OPLÆG FRA TOMMY LANGHOFF

OPLÆG TIL

CALL IMPROVEMENT WEB

9.1.1 INDLEDNING

Ledelse og udvikling af et kundekontaktcenter (call center også brugt) er ved at udvikle sig til en egen fagdisciplin. Det er dog stadig en ung disciplin med kun lidt fælles referenceramme mellem virksomhederne for, hvorledes problemstillinger angribes.

Centrene bruger og har traditionelt brugt meget tid på alt det som ligger bag den enkelte kundeservice medarbejder, såsom IT systemer, processer, produktviden osv. – men der i mod fortsat alt for lidt opmærksomhed på de x minutter, hvor slaget virkelig slås – nemlig mens kundeservice medarbejderen har kunden i røret (eller mailen mv.) og her afgør om det bliver en god eller dårlig oplevelse.

Dette misforhold er man ved at få øjnene op for, men typisk ved ledelsen ikke, hvorledes de skal gribe problemstillingen an.

Call Improvement skal give centrene et værktøj til at systematisere og styre en øget fokus på individuel kvalitetsudvikling jf. ovenfor. Både i det daglige praktiske arbejde mellem leder og medarbejder, hvor resultaterne diskuteres og aktiviteter og uddannelse aftales, men også for kundeservice chefen som har behov for en præcis omgang med kvalitetsdata og kvalitetsudvikling, således at han/hun kan styre organisationen og rapportere ud ad til.

Et kundecenter er en meget dynamisk enhed, som ofte er langt foran resten af organisationen hvad angår risikovillighed, beslutningshastighed mv. De er ofte en enhed i organisationen med lav fokus og prestige. Der er altså behov for at kunne agere uden for meget afhængighed af resten af organisationen. En helt afgørende ting ved konceptet og systemet er derfor, at det skal være "easy to go" – ingen lokal installation, minimum af opsætning, ingen opstartsomkostninger osv. Altså maksimal uafhængighed af andre dele af organisation

En systematisk og effektiv individuel kvalitetsudvikling vil med meget stor sandsynlighed give markante ændringer i form af:

Bedre kundeoplevelse og dermed image, salg mv.

Reducerede omkostninger per kundehenvendelse i form af kortere samtaletider

Reducerede omkostninger i form af mere præcis uddannelse

Det er centralt for konceptet og systemets salgsmæssige succes at:

Det er "easy to go"

At systemet bibringer den enkelte proces og arbejdsopgave ekstra værdi i form af hastighed, kvalitet eller andet

Brugervenligt og intuitivt nemt at anvende

Sikkerhedsmæssigt helt upåklageligt hvad angår beskyttelse af persondata

At der er et sprog-lag som muliggør salg på andre markeder end Danmark

9.1.2 SYSTEMET

9.1.2.1 Overordnet struktur

Call Improvement Web (CIW) er en del af et koncept, som ud over systemet også består af:

- Uddannelsespakker for medarbejdere og ledere
- En IVR løsning leveret af Consorte A/S til måling af kundernes tilfredshed med det enkelte kald (med integration til CIW jf. senere)

CIW består af følgende moduler/menuer/områder (i tilfældig rækkefølge):

1. Samlyt
2. Uddannelse & vejledning
3. Market info
4. Questionnaire
5. IVR
6. Rapport & analyse
7. Administration

Jeg vil i det følgende beskrive hvert punkt med følgende struktur:

- Formål
- Funktionalitet og proces

Rapportering og analyse for de enkelte moduler foreslås samlet under eget samlet punkt Rapport & analyse.

9.1.3 MODUL 1 – SAMLYT

Formål:

Et værktøj til praktisk styring af gennemførelse af samlyt, lagring af data og opfølgning på samlyt

Funktionalitet og proces:

Processen i en samlyt:

1. Lederen printer en samlyt protokol* fra systemet, et feedback skema* og evt. rapporter mv. fra tidligere samlyt som forberedelse til seancen
2. Lederen lytter til x antal samtaler ved siden af eller på afstand af medarbejderen
3. Lederen scorer de enkelte faser i en samtale på en skala 1-7 jf. protokollen og noterer scoringerne på protokollen sammen med oplysninger om:
   - Medarbejderens navn og/eller bruger ID i systemet
   - Kaldstype* ud fra prædefineret liste af muligheder for denne kundeprofil
   - Antal kald der er lyttet på
   - Dato for samlyt
4. Lederen gennemfører en feedback samtale med medarbejderen, hvor observationer og scoringer gennemgås og diskuteres. Desuden udfyldes feedback skemaet jf. nedenfor
5. Lederen taster (eller scanner) scoringerne ind i et indtastningsbillede og data lagres i databasen. Tilsvarende for feedback skemaet, som dog (så vidt jeg lige nu kan bedømme) kun behøves gemt som pdf.fil eller lignende
6. Lederen eller medarbejderen kan løbende lave rapporter og analyse for at følge udviklingen eller åbne seneste feedback skema og lave opfølgning på de gennemførte aktiviteter.
7. Det er tid til samlyt igen og processen starter forfra
8. Med jævne mellemrum gennemfører lederne en såkaldt ”kvalitets kalibrering”, dvs. de sætter sig sammen med optagede samtaler og scorer disse hver for sig og diskuterer efterfølgende dette, sådan at deres opfattelse af god kvalitet ”kalibreres”.
   Under punktet/modulet Samlyt, skal lederne kunne registrere

gennemførte kalibreringer med angivelse af deltagere (ledernes navn og/eller bruger ID), antal lyttede samtaler og dato

Protokollen:

Simpelt eksempel, kun til forklaringsbrug:

Fase	Scoring	Vægt	Samlet
Kontakt fasen - Velkomst - Præsentation Samlet	 5 4 4,5	 30%	 1,35
Behovsafdækning - Spørgeteknik - Behov afdækket ok	 3 3		
Løsning			
Osv.			

NB! Systematikken omkring vægte, procent fordeling af scorede point mv. er ikke færdig udviklet. Er kun til forklaringsbrug!

Vil når færdigudviklet udtrykke kvaliteten for hver sektion og samlet i % af det optimale.

Hver sektion vil have et antal fast definerede punkter gældende for alle kunder (de generelle) og så skal der være mulighed for at definere 2-3 punkter specifik for kunden.

Kunden skal kunne ændre vægtene – sker i modulet Administration – Samlyt

Man kunne forestille sig at nogle centre ville ønske muligheden for at indscanne de udfyldte protokoller i stedet for indtastning. Lad os diskutere hvad det vil betyde af udviklingsarbejde.

Feedback skemaet:

Skemaet er ikke færdig udviklet, men vil indeholde oplysningerne:

- Navn og/eller bruger ID på hhv. medarbejder og leder
- Dato for samlyt
- Lederens konklusioner og anbefalinger
- Medarbejderens konklusioner og anbefalinger
- Afledte aftalte aktiviteter (uddannelse eller andet) med ansvar og deadlines
- Tidspunkt for næste samlyt

Kaldstype:

Medarbejderen kan have kompetencer på forskellige områder, f.eks. produktområde A, B eller C. Det er vigtigt efterfølgende at kunne følge op på kvaliteten i forhold til det enkelte produktområde.

Som en del af definitionen af kundeprofilen skal der laves en opsætning af kaldstyper for kunden, således at der vælges mellem en prædefineret liste.

9.1.4 MODUL 2 – UDDANNELSE & VEJLEDNING

Formål:

Under dette punkt/modul tilbydes ledere og medarbejdere materialer, downloads, links, tilbud, manualer, vejledninger mv. til arbejdet med individuel kvalitetsudvikling.

Funktionalitet:

Dette modul lagrer ingen data, men stiller information til rådighed på forskellig vis.

Underpunkter:

1. Materialer: Downloads af forskellige materialer i MS formater, som anvendes på uddannelserne som er knyttet til konceptet
2. Links: Links til websites med interessant og inspirerende information med relation til temaet
3. Uddannelse: Beskrivelse af de aktuelle uddannelsestilbud fra TLC og partnere, som knytter sig til temaet
4. Vejledning: Bruger vejledning til CIW, vejledning for gennemførelse af samlyt inkl. etiske regler, persondata beskyttelses regler og vejledning, vejledning for ”samlyt kalibrering”*, vejledning for feedback etc.

9.1.5 MODUL 3 – MARKET INFO

Formål:

En typisk problemstilling i kundeservice er, pga. den store dynamik og foranderlighed, at man bliver spurgt om f.eks: ”Hvor mange af de kald som kommer, handler om dit eller dat…?”

I stedet for den gængse men ineffektive pindestatistik på papir, skal dette kunne gøres i systemet.

Funktionen vil også i stedet / samtidigt for denne ad-hoc og midlertidige registrering kunne anvendes til en fast og løbende klassificering af alle kald efter type, kundetype etc.

Funktionalitet:

Proces:

1. Medarbejderen afslutter et kald (eller endnu bedre samtidig med kaldet)
2. Medarbejderen registrerer type etc. jf. kundens valg af anvendelse direkte i CIW og data lagres i databasen

For at optimere hastigheden i kundehåndteringen, skal medarbejderen kunne have dette skærmbillede åbent på skærmen hele arbejdsdagen. Skal altså også kunne gå ind andre steder i CIW uden at dette skærmbillede lukkes.

9.1.5.1 Registrering:

Eksempel på struktur:

1 Klagesag
   i. Faktura klage
   ii. Service klage
   iii. etc.

2 Købshenvendelse
   i. salg type A
      1. Afsluttet salg
      2. Ikke afsluttet salg
   ii. salg type B

3 Service forespørgsel
   i. forespørgsel type A
   ii. forespørgsel type B

Jeg mener at 3 niveauer er tilstrækkeligt.

En hurtig og nem registrering er afgørende. 2 sek. ekstra hvis man tager 100 kald hver dag bliver til mange timer på et år…

9.1.6 MODUL 4 – QUESTIONNAIRE

Formål:

Et kundecenter har løbende brug for at stille medarbejderne spørgsmål, primært i to anledninger:

- Gennemførelse af interne analyser, f.eks. medarbejder tilfredshedsundersøgelser
- Gennemførelse af tests, f.eks. test af medarbejderens aktuelle viden, hvilken jo er afgørende for kvaliteten

Modulet Questionnaire skal gøre det muligt hurtigt og nemt at oprette, ”release” og gennemføre en sådan test eller analyse.

Funktionalitet:

Proces:

1. Selve spørgeskemaet er af multiple choice typen og oprettes under Administration – Questionnaire jf. senere.
   Her skal lederen kunne definere et spørgeskema med op til 75 spørgsmål med op til 4 svarmuligheder til hver.

   Jeg forestiller mig at opretteren her angiver om det er af typen ”test” eller ”analyse”. Såfremt det er af typen ”test” skal det korrekte svar angives i oprettelsen.

   Angivelse af tidsfrist for besvarelse.

   Angivelse af hvem der skal gennemføre testen / analysen (individ, gruppe, produkt, center jf. brugerdata)

   Angivelse af om der anonymitet på testen/analysen eller ikke

2. Opretteren releaser testen / analysen

3. Testen / analysen bliver tilgængelig under Questionnaire med angivelse af tidsfrist og hvem der skal gennemføre (individer, grupper, produkt, center)

4. Medarbejderen udfylder/besvarer og godkender – data lagres i databasen.
   Hvis ikke anonymitet på testen / analysen angives navn og/eller

bruger ID

5. Ved Questionnaire (test) får medarbejderen en tilbagemelding på skærmen på rigtige og forkerte svar.
   Ved Questionnaire (analyse) et "tak for din besvarelse" på skærmen

6. Lederen kan følge fremdriften i besvarelserne (i % hvis anonymitet og med navne hvis ikke anonymitet) og udskrive rapporter etc. under Rapport & Analyse – Questionnaire jf. senere

7. Lederen giver medarbejderen en tilbagemelding inkl. de rigtige svar på evt. fejl – ved Questionnaire (test)

9.1.7 MODUL 5 – IVR

Formål:

IVR (Interactive voice respons) løsningen giver mulighed for at spørge kunderne direkte efter afslutning af det enkelte kald omkring deres kundeoplevelse.

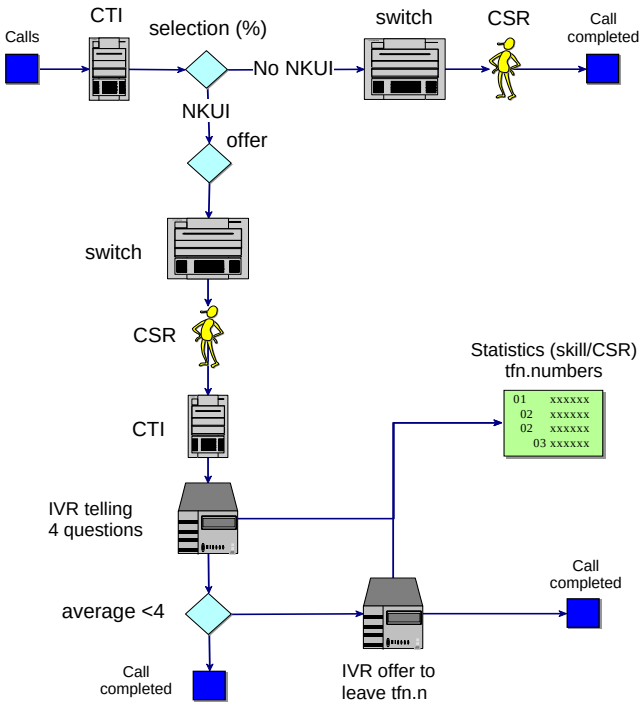
Funktionalitet:

IVR løsningen etableres af Consorte A/S, som udvikler telefoniløsninger baseret på en stor IN-løsning og IVR setup.

Kaldene routes ind i gennem Consortes IVR, som stiller det indledende spørgsmål om, hvorvidt kunden ønsker at deltage, herefter hen til medarbejderen. Såfremt kunden har sagt ja, holder IVR fast i kaldet, registrerer medarbejderens ID og tager kaldet tilbage til IVR efterfølgende og stiller de ønskede spørgsmål. Kunden svarer ved tastetryk på en defineret skala.

IVR kan stille et ekstra spørgsmål, f.eks. bede om navn og telefonnummer, hvis f.eks. gennemsnittet af kundens besvarelse er under et bestemt niveau.

Kan illustreres således:



Løsningen vil være et tilkøb og håndteres teknisk af Consorte.

Snitfladen til CIW er en import af data fra Consortes database til CIWs database.
Hvorledes dette kan ske må afklares mellem teknikere og udviklere fra NextUs og Consorte.

Følgende data vil komme med over fra IVR databasen per besvarelse:

- Bruger ID
- Besvarelses tidspunkt
- Besvarelserne på spm. 1 til X, udtrykt i % af det optimale

Følgende generelle informationer skal med over fra IVR databasen:

- Antal besvarelser i % af de adspurgte
- Antal accepter i % af de adspurgte (differencen er de som sagde ja, men alligevel lagde på inden besvarelsen)

Spørgsmålenes ordlyd, skala og udvælgelses-procent skal defineres i Kundeprofilen i Modulet Administration jf. senere

9.1.8 MODUL 6 – RAPPORT & ANALYSE

Formål:

Mulighederne i Rapport & analyse tjener 2 overordnede formål:

1. At klæde den enkelte mellemleder på til sin coaching rolle i udviklingen af den enkelte medarbejder
2. Mangement informations system (MIS), der klæder både mellemledere og kundeservicechefen på i relation til deres planlægning, målopfølgning, budgetlægning, rapportering mv.

Med disse muligheder bliver individuel kvalitet et integreret mål og arbejdsområde og ikke bare et buzzword alle taler om, men ingen ved hvad der menes med og hvor man står.

Funktionalitet:

Jeg er ikke helt afklaret med strukturen af de forskellige rapport og analyse muligheder – altså hvordan de skal grupperes, undermenuer etc. Det må vi arbejde videre på. Nedenstående er altså en relativ vilkårlig oplistning af kravene. Jeg kan godt se at nogle af forklaringerne er langhårede, men så må jeg løbe dem igennem med udviklerne på tavlen. Der er givet vis også rapporter som vil dukke op undervejs i udviklingsprocessen som relevante.

Man kunne måske forestille sig at alle rapporterne genereres direkte i Excel inkl. grafik mv. Alternativt skal data i hvert fald enkelt kunne eksporteres til Excel således at der kan arbejdes videre med data om ønskeligt.

Flest mulige rapporter præsenteres med grafik og data.

9.1.8.1 MIS

1. Udvikling for den scorede kvalitet udtrykt i % over et tidsinterval opdelt i periodetyper (måneder eller kvartaler af hensyn til statistisk sikkerhed) med mulighed for stillingtagen til disse parametre:
   - Vælge score for den samlede protokol, for protokol sektion, enkelt punkt i protokollen eller alle 3 dele
   - For individ, gruppe, afdeling, funktion, center
   - For en eller flere kaldstyper

   Vises grafisk og med data.

2. % fordeling på grupperinger af kvalitetsniveau (score for den samlede protokol) (f.eks. 10% af de valgte individer ud fra søgningens kriterier ligger på kvalitetsniveau 60-70%, 12% på 70-80% etc.). Mulighed for stillingtagen til parametrene:
   - Gruppering ud fra brugerdata (individ, gruppe, afd. etc.)
   - Tidsinterval
   Vises grafisk og med data.

3. Top 3 / bund 3 enkelt scoringspunkter i protokollen for en gruppering (jf. brugerdata) og tidsinterval

4. Analyse værktøj på tværs af modulerne Samlyt, Questionnaire (test) og IVR hvor parametrene kan kombineres. F.eks. hvis en gruppering (individ, gruppe, afd. etc.) har under X% i Questionnaire for et valgt tidsinterval hvad er så deres score i % på Samlyt sektionen ”Løsning”).
   Kaldes vel betingede søgninger med mulighed for og/eller kombinationer (??)

5. Antal registreringer per parametre i Market Info for en valgt tidsperiode

6. ”Kundechef rapport”: Simpel grafisk fremstilling af samlede kvalitet-% for hhv. Samlyt, Questionnaire (test) og IVR i en valgt tidsperiode

7. Tilsvarende pkt. 6 med udviklingen over et tidsinterval i valgte periodetyper (måneder eller kvartaler)

8. IVR: Antal scoringer gennemført af kunderne, antal forespørgsler fra IVR om deltagelse og gennemførte i % af forespurgte. For valgt tidsinterval.

9. IVR: Den samlede score samt scoren per spørgsmål i IVR for et valgt tidsinterval med stillingtagen til en gruppering (individ, gruppe, afdeling etc. jf. brugerdata)

10. Tilsvarende pkt. 9 med udviklingen over et tidsinterval i valgte periodetyper (måneder eller kvartaler)

11. Questionnaire (test): % fordeling af den valgte gruppering (individ, gruppe, afd. etc.) over en gruppering af score i % (f.eks. 30% af den valgte gruppering har over 90% korrekte, 20% har 75-90% korrekte svar etc.). Udgangspunkter er en specifik released test.

12. Questionnaire (test): Top 5 forkerte svar for en gruppering på en specifik released test.

13. Questionnaire (test): Udvikling i et tidsinterval i % rigtige svar (samlede test) på en valgt gruppering. Forudsætter ens test type for at give mening (f.eks. en kontinuerlig test i aktuel produktviden)

14. Questionnaire: Status på besvarelserne på en specifik test / besvarelse ift. den gruppering som den er released til. Hvis anonymitet: I %, hvis ikke: Med navne på de som mangler.

15. Questionnaire (analyse): % fordeling på de enkelte svarmuligheder per stillet spørgsmål for en valgt gruppering. På en specifik analyse.

9.1.8.2 Coaching

1. Samlet individuel kvalitetsrapport for et valgt individ, som viser resultaterne for et valgt tidsinterval for hhv. Samlyt, Questionnaire (test) og IVR. Samlyt vises samlet og per sektion. IVR vises samlet og per spørgsmål. Questionnaire vises samlet.
   Rapporten indeholder desuden udtryk for "pålidelighed" i form af antal scoringer i Samlyt i perioden, antal IVR scoringer og antal Questionnaire (test)
   Rapporten indeholder kvalitative anbefalinger til lederen til hvorledes udviklingen af denne medarbejder bør gribes an. Sker i form af sammensatte tekststrenge ud fra kombinationer og niveauer for parametrene i Samlyt, Questionnaire (test) og IVR. Ambitiøst – men jeg tror det er muligt!

2. Historisk oversigt for et individ over hvornår samlyt og Questionnaire (test og analyse) er blevet gennemført og med hvilke resultater (for samlyt : samlet og per sektion, for Questionnaire (test): samlet resultat i %, for Questionnaire (analyse): blot dato for gennemførelse, hvis ikke anonymitet)

3. Søge oversigt over lagrede feedback skemaer for et individ, og herefter vælge den ønskede samt mulighed for at rette i den seneste (for evt. opdatering i status på aftale aktiviteter). Alle skemaer skal kunne printes.

4. Oversigt over en leders gennemførte samlyt i et tidsinterval: Samlede antal samlyt i perioden og antal per medarbejder der er gennemført samlyt med.

5. Søge oversigt over tidligere indtastede protokoller fra samlyt frem så der dannes en liste. Herefter klikke på den relevante for at kunne se detaljerne og evt. printe. Skal ikke kunne rette når først data er gemt.

9.1.9 MODUL 7 - ADMINISTRATION

Formål:

Administrations modulet skal samle al opsætning mv. for kunden selv samt indeholde intern administration for os selv ifm. faktureringsgrundlag etc.

Udgangspunktet er at kunden selv skal stå for at slette/oprette brugere, ændre parametre opsætninger etc. når først vi indledningsvis én gang har oprettet kundeprofilen og brugerne ved opstart. Vi skal for alt i verden undgå en masse løbende administration og support på CIW, skal være så selvkørende for os som muligt.

Funktionalitet:

9.1.9.1 IVR

- Mulighed for definition af "selection %", dvs. andelen af indgående kald som skal spørges om deltagelse. CIW kan ikke ændre direkte i IVR´en som ligger hos Consorte men generere en request til Consorte. Vil være formodentlig være forbundet med et gebyr fra Consorte, hvilket kunden skal oplyses om når han/hun sender denne request

- Definition af spørgsmålene i IVR. Sker igen ved at sende en request til Consorte med ønsker til ændringer i spørgsmålene. Igen mod gebyr som kunden oplyses om.

- Definition af skala for kundens bedømmelse. Igen via request og mod gebyr.

Alternativt opgiver vi under Administration – IVR et support nummer hos Consorte samt en prisliste for ændringer.

9.1.9.2 Samlyt

- Definition af de "frie" spørgsmål i protokollen under de respektive sektioner. Der skal rettighedsbegrænsning på dette og kunden oplyses om, at det har effekt på muligheden for at sammenligne fra før og efter ændringen. Sker ved skærm pop-up med advarsel!

- Definition af vægtene på de enkelte spørgsmål og sektioner. Igen rettighedsbegrænsning og pop-up med advarsel omkring sammenligning af data.

- Definition af kaldstyper, dvs. vedligeholdelse af listen over mulige kaldstyper i protokollens felt for "Kaldtype"

9.1.9.3 Market info

- Opsætning af parametrene på de 3 niveauer jf. tidligere beskrivelse

9.1.9.4 Questionnaire

- Skabe en ny test eller analyse til Release: Definere spørgsmål og svarmuligheder, definere tidsfrist for besvarelse, sende selve releasen, definere evt. anonymitet

9.1.9.5 Generelt

- Brugerdata: Brugeren skal kunne oprettes af kunden med følgende parametre, hvoraf nogle vil være tvungne andre frivillige:
  - Navn
  - Bruger ID (evt. tildelt af systemet?)
  - Gruppe (ud fra defineret liste under Kundeprofilen)
  - Afdeling (niveau over gruppe) (ud fra defineret liste under Kundeprofilen)
  - Kundecenter (kunden kan have flere centre) (ud fra defineret liste under Kundeprofilen)
  - Produktområde (samme gruppe kan have flere produktområder) (ud fra defineret lise under Kundeprofilen)
  - Ansættelsesdato i defineret format
  - Funktion (fritekst)

- Brugere skal kunne slettes af kunden selv. Ved sletning skal kunden oplyses om at "brugeren er registreret som aktiv i måned X, licensen vil derfor være opsagt per [dato]".

- Kundeprofil: Skal der tænkes lidt mere over, men jeg forestiller mig følgende parametre:
  - Virksomhedsdata inkl. CVR nr.
  - Kontaktperson
  - Faktureringsadresse
  - Liste over Grupper
  - Liste over Afdelinger
  - Liste over Centre

- Liste over Produktområder
- Prisaftale
- Evt. forhandler som har solgt til kunden (af hensyn til provisionsafregning

Intern adm.

- Liste over brugere (med navne) per kunde som indenfor en defineret kalendermåned har været oprettet i systemet på noget tidspunkt

- Generering af faktureringsgrundlag: Antal brugere x pris jf. prisaftale i kundeprofilen. Præsenteres direkte som faktura i TLC format klar til at blive gemt i MS Word format hos TLC for påsætning af fakturanr. mv. og forsendelse.

- Vedligeholdelse af en liste over forhandlere, som har adgang til at sælge systemet. Skal inkludere forhandlerens navn, kontaktperson samt provisionsaftalen.

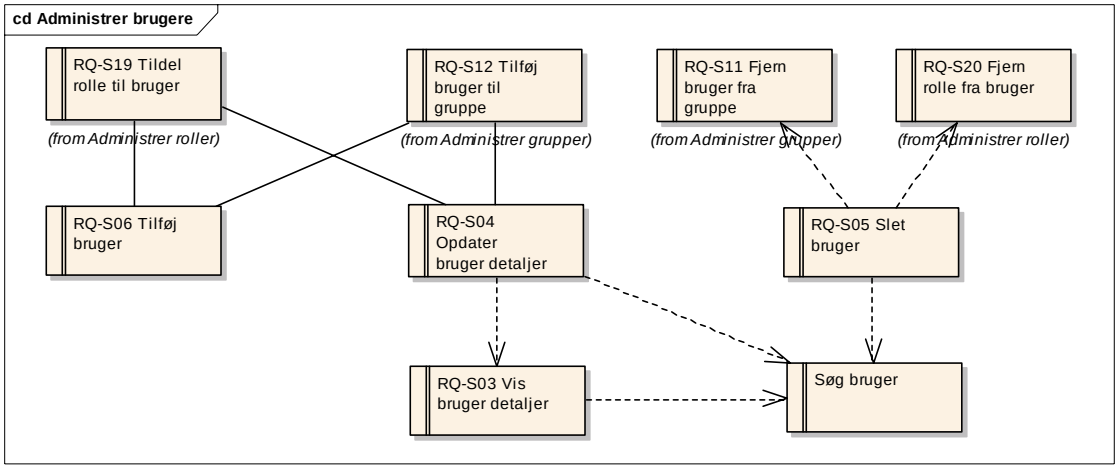
- Generering af grundlag for provisionsafregning: Antal kunder med forhandleren påført kundeprofilen. For hver kunde faktureringsgrundlaget for kunden x provisionssatsen samt en sum for forhandleren.

9.2.1 KRAVSPECIFIKATION SYSTEM

9.2.1.1 Adgangs kontrol.

Navn	Beskrivelse
RQ-S01 Valider bruger	Det skal være muligt at validere en bruger i systemet på baggrund af brugernavn og password.
RQ-S02 Autoriser bruger	Det skal være muligt at checke hvor vidt en bruger har tilladelse til at udføre en bestemt handling ud fra de roller brugeren er tildelt.

9.2.1.2 Administrer brugere.

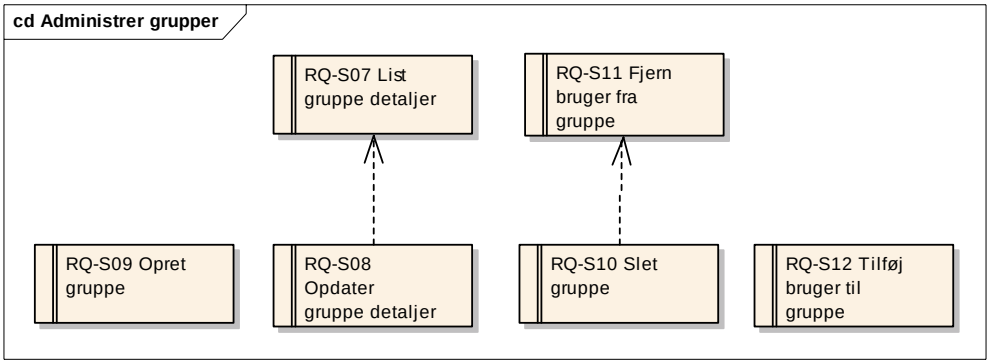


Figur: Administrer brugere

Navn	Beskrivelse
RQ-S03 Vis bruger detaljer	Det skal være muligt at få vist en brugers detaljer.
RQ-S04 Opdater bruger detaljer	Det skal være muligt at opdatere en brugers detaljer.
RQ-S05 Slet bruger	Det skal være muligt at slette en bruger fra systemet. Når brugeren slettes skal denne fjernes fra evt. grupper.
RQ-S06 Tilføj bruger	Det skal være muligt at tilføje en bruger til systemet. En bruger har som minimum følgende detaljer - Navn - Stilling - E-mail - Telefon nr. - Grupper (Hvilke grupper brugeren er medlem af). - Roller (Hvilke roller brugeren er tildelt).

Navn	Beskrivelse
Søg bruger	Det skal være muligt at søge på en bruger

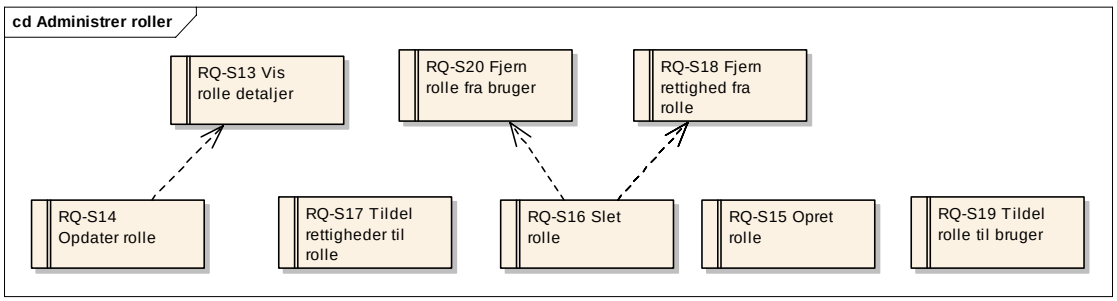
9.2.1.3 Administrer grupper.



Figur: Administrer grupper

Navn	Beskrivelse
RQ-S07 List gruppe detaljer	Det skal være muligt at få vist detaljerne på en gruppe.
RQ-S08 Opdater gruppe detaljer	Det skal være muligt at opdatere en gruppes detaljer.
RQ-S09 Opret gruppe	Det skal være muligt at oprette en gruppe der kan indeholde brugere i systemet. En gruppe har følgende detaljer: - Navn - Beskrivelse - Medlemmer (Brugere i systemet)
RQ-S10 Slet gruppe	Det skal være mulig at slette en gruppe fra systemet. Når en gruppe slettes, fjernes tilhørsforholdet til gruppen fra de brugere der evt. måtte være i den.
RQ-S11 Fjern bruger fra gruppe	Det skal være muligt at fjerne en bruger fra en gruppe.
RQ-S12 Tilføj bruger til gruppe	Det skal være muligt at tilføje en bruger til en gruppe.

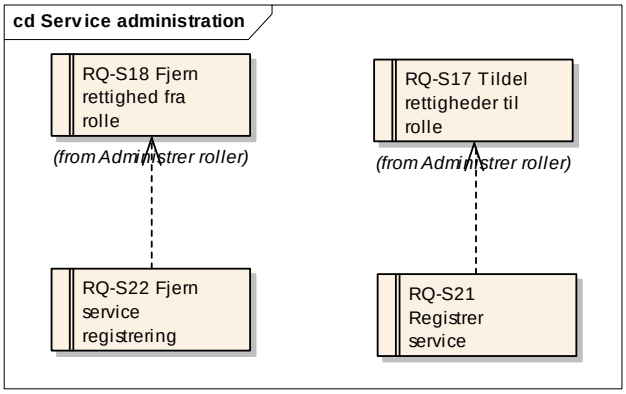
9.2.1.4 Administrer roller.



Figur: Administrer roller

Navn	Beskrivelse
RQ-S13 Vis rolle detaljer	Det skal være muligt at få vist en rolles detaljer.
RQ-S14 Opdater rolle	Det skal være muligt at opdatere en rolles detaljer.
RQ-S15 Opret rolle	Det skal være muligt at oprette en rolle i systemet. En rolle indeholder følgende detaljer: - Navn - Beskrivelse - Medlemmer (bruger /grupper der har fået tildelt rollen) - Hvilke rettigheder rollen giver medlemmer
RQ-S16 Slet rolle	Det skal være muligt at slette en rolle fra systemet. Når en rolle slettes fjernes tilhørsforholdet til rollen fra de brugere/grupper den evt. måtte være tildelt rollen.
RQ-S17 Tildel rettigheder til rolle	Det skal være muligt at tildele en rolle rettigheder til at udføre operationer, som de i systemet registrerede services stiller til rådighed.
RQ-S18 Fjern rettighed fra rolle	Det skal være muligt at fjerne tildelte rettigheder fra roller.
RQ-S19 Tildel rolle til bruger	Det skal være muligt at tildele en rolle til en bruger
RQ-S20 Fjern rolle fra bruger	Det skal være muligt at fjerne en rolle fra en bruger der tidligere er blevet tildelt denne

9.2.1.5 Service administration.



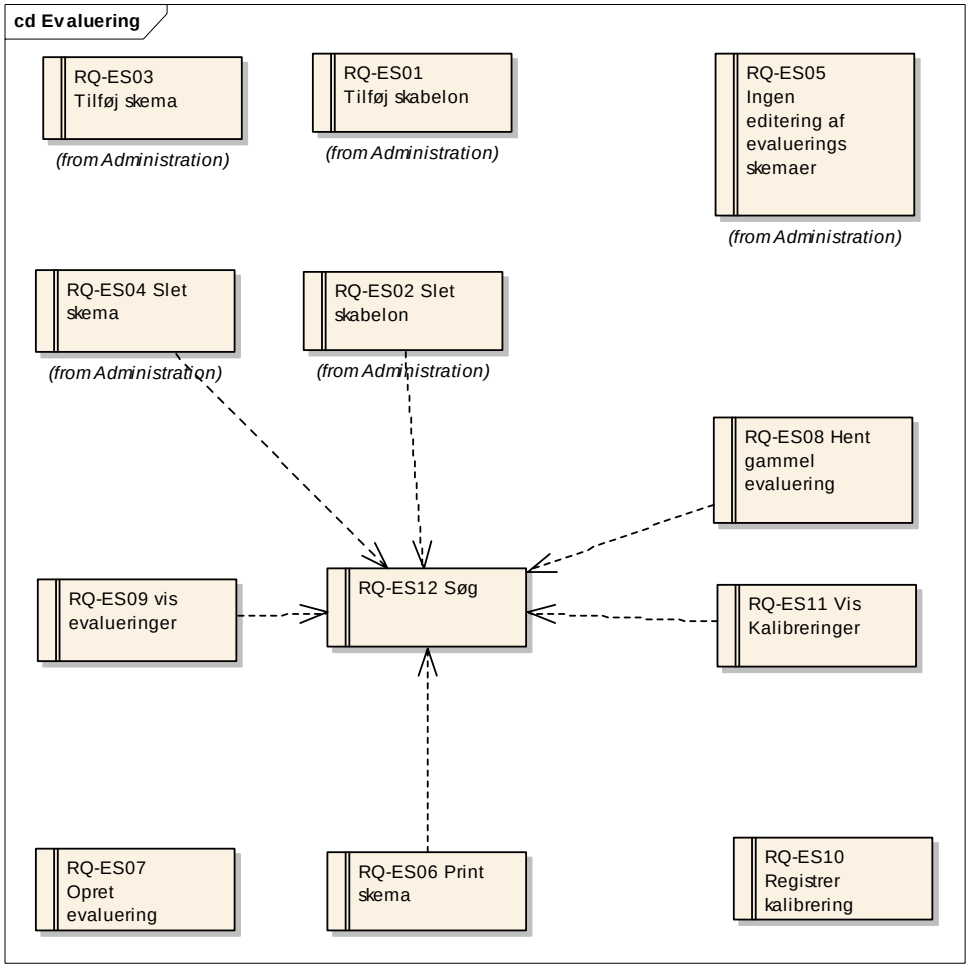
Figur: Service administration

Navn	Beskrivelse
RQ-S21 Registrer service	Det skal være muligt at registrere en service i systemet, således dennes funktionalitet kan benyttes. Ved registreringen skal følgende gemmes i systemet: - Navnet på services - Forbindelses metode (http, tcp, named pipes etc.) - Servicens adresse - Liste af operationer servicen tilbyder.

Navn	Beskrivelse
RQ-S22 Fjern service registrering	Det skal være muligt at fjerne en service registrering fra systemet. Når services fjernes fra systemet, skal alle roller med rettigheder til at udføre operationer fra services, opdateres således at disse rettigheder fjernes.

9.2.2 KRAVSPECIFIKATION EVALUERINGS SKEMAER

Requirements Model.



Figur: Evaluering

9.2.2.1 Administration.

Navn	Beskrivelse
RQ-ES01 Tilføj skabelon	Det skal være muligt at oprette skabeloner i systemet. Skabelonerne skal kunne være udgangspunkt for oprettelse af nye skemaer. En skabelon skal tilhøre en bestemt type, som vælges fra en prædefineret liste. Der skal som

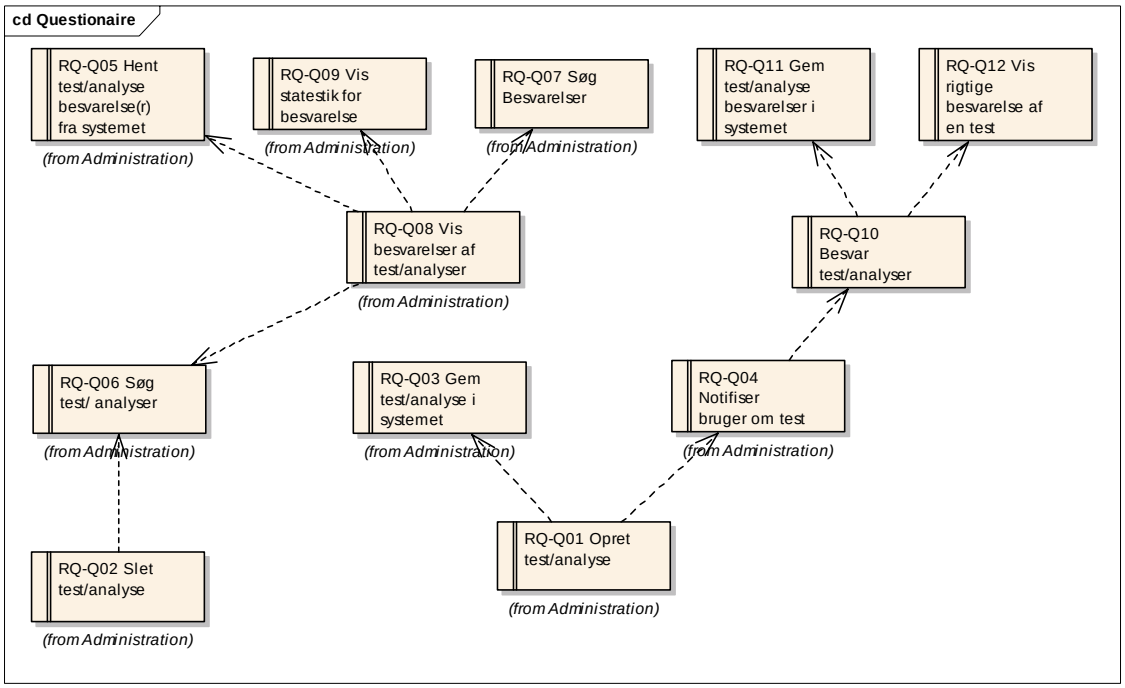
Navn	Beskrivelse
	minimum kunne oprettes skabeloner af typerne: - Protokol - Feedback-skema
RQ-ES02 Slet skabelon	Det skal være muligt at slette en skabelon i systemet.
RQ-ES03 Tilføj skema	Det skal være muligt for en System-administrator/ team-leder at tilføje et skema til brug ved evaluering. Det skal være muligt at oprette skemaerne på baggrund af skabeloner allerede oprettet i systemet. Der skal være muligt at oprette skemaer af typen Protokol og Feedback, som anvendes til evalueringen af medarbejderen. Protokollen skal indeholde: - Navn - X antal punkter som medarbejderen skal bedømmes på: - tekst - scoring - relativ vægt Feedbackskemaet skal indeholde følgende: - Navn - X antal punkter med feedback punkter. Når et skema oprettes ud fra en skabeloner, skal der kunne oprettes x antal valgfrie felter, udover hvad skabelonen indeholder
RQ-ES04 Slet skema	Det skal i systemet være muligt at slette et skema.
RQ-ES05 Ingen editering af evaluerings skemaer	Når et evalueringsskema er oprettet i systemet, skal det ikke være muligt at rette i dette.

9.2.2.2 Generel funktionalitet

Navn	Beskrivelse
RQ-ES06 Print skema	Lederen skal kunne printe et af de i systemet definerede evalueringsskemaer/protokoller. Dette af hensyn til at det kan udfyldes i hånden, for senere at kunne blive skrevet ind i systemet. Det skal være muligt at printe skemaer gemt i allerede foretagende evalueringer.
RQ-ES07 Opret evaluering	En leder skal kunne udføre en evaluering. Der skal deraf kunne oprettes en evaluering i systemet som består af en udfyldte protokol og et feedback skema. Ud over de udfyldte skemaer skal der for evalueringen desuden gemmes følgende: - Hvilken leder har udført evalueringen - Hvilken medarbejder blev evalueret - Tidspunkt for evaluering

Navn	Beskrivelse
	- Type (vælges ud fra en prædefineret liste) - Antal seancer der er lyttet til.
RQ-ES08 Hent gammel evaluering	Det skal være muligt at hente tidligere foretagende evalueringer frem af systemet igen.
RQ-ES09 vis evalueringer	Det skal være muligt at hente og vise evalueringer fra systemet. Herunder tilhørende information og skemaer
RQ-ES10 Registrer kalibrering	Fra tid til anden foretager lederne en samlet kalibrering hvor der gennemlyttes et antal samtaler. Disse scores individuelt af deltagerne i kalibreringen. Efterfølgende evalueres disse med henblik på en kalibrering. Når disse kalibreringer er fortaget skal de gemmes i systemet. Til en registrering af en kalibrering skal der gemmes følgende: - Deltagere i kalibreringen. - Dato for kalibrering. - Antal lyttede samtaler. - Noter
RQ-ES11 Vis Kalibreringer	Det skal være muligt at få vist de foretagede kalibreringer, med henblik på at få information om en tidligere kalibrering.
RQ-ES12 Søg	Det skal i servicen være muligt at fortage søgning på følgende: - Skemaer - Protokoller - Evalueringer - Kalibreringer Søningen skal eksempelvis kunne fortages på baggrund af følgende kriterier: - Bruger - Skema/protokol type - Tidsperiode for oprettelse

Requirements Model.



Figur: Questionaire

9.2.3.1 Administration.

Navn	**Beskrivelse**
RQ-Q01 Opret test/analyse	Det skal være muligt for en leder at oprette en test/analyse. En test/analyse skal indeholde følgende informationer: - Navn på test/analyser - Type: Test eller analyse - Dato for senest besvarelse - Angivelse af hvem der skal have mulighed for at gennemføre testen (bruger(e), gruppe(r)) - Angivelse af om der er tvungen besvarelse - Angivelse af hvem der har oprettet testen/analysen. - Max 75 spørgsmål med hver 4 svar muligheder skal angives - Angivelse af om besvarelse er anonymt. Hvis det er en test skal de rigtige svar på hvert enkelt spørgsmål ligeledes angives.

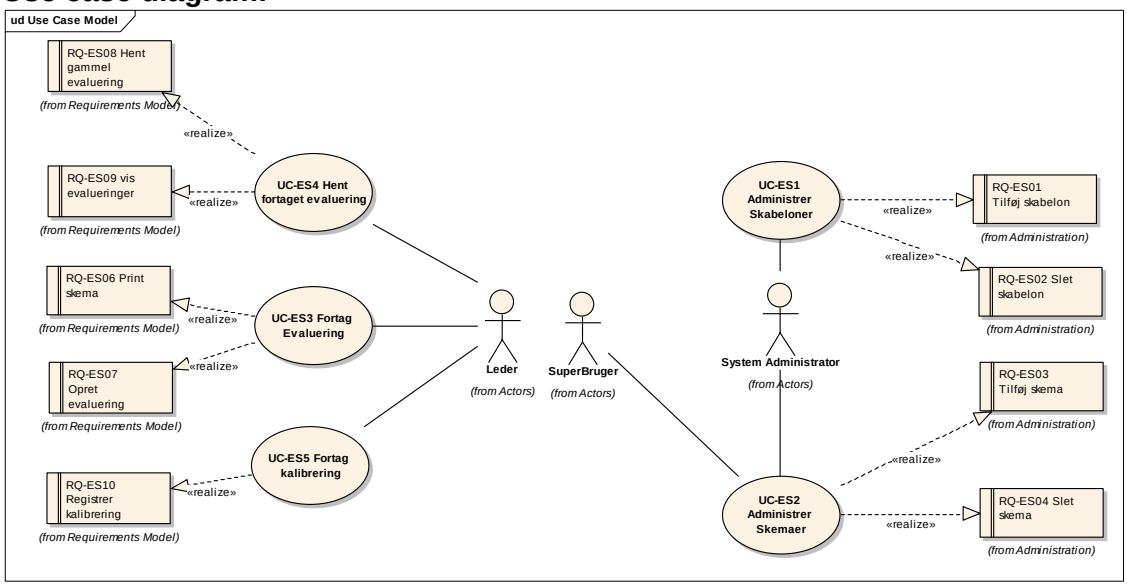
Navn	Beskrivelse
RQ-Q02 Slet test/analyse	Det skal være muligt at slette en test/analyse der er oprettet i systemet
RQ-Q03 Gem test/analyse i systemet	En test/analyse skal kunne gemmes i systemet.
RQ-Q04 Notificer bruger om test	Hvis en medarbejder tildeles en test eller analyse skal medarbejderen notificeres om dette. Dette kan enten gøres ved mail eller en besked på medarbejderens start side. Der er intet specifikt krav om hvorledes denne notificering skal foregå.
RQ-Q05 Hent test/analyse besvarelse(r) fra systemet	Det skal være muligt at hente én eller flere test/analyse besvarelser fra systemet.
RQ-Q06 Søg test/ analyser	Det skal være muligt at søge tests og analyser i systemet
RQ-Q07 Søg Besvarelser	Det skal være muligt at søge besvarelser i servicen på passende kriterier f.eks. - Tests / Analyser - Brugere - Grupper
RQ-Q08 Vis besvarelser af test/analyser	Det skal være muligt for lederen at se følgende om udgivende test/analyser: - Hvor mange der har besvaret, faktisk værdi og procent af de medarbejdere den er tildelt. - Hvis ikke anonym, hvilke medarbejdere der har besvaret - Hvilket svar der er afgivet til de enkelte spørgsmål.
RQ-Q09 Vis statestik for besvarelse	Servicen skal kunne generere samlet information og statistik for en foretaget test eller analyse.

9.2.3.2 Generel funktionalitet

Navn	Beskrivelse
RQ-Q10 Besvar test/analyser	Det skal være muligt for medarbejdere at besvare tests og analyser som er tildelt dem.
RQ-Q11 Gem test/analyse besvarelser i systemet	Når en medarbejder har besvaret en test/analyse skal besvarelsen gemmes i systemet
RQ-Q12 Vis rigtige besvarelse af en test	Efter endt udfyldelse af en test skal medarbejderen præsenteres med de rigtige svar til testen. Herunder: - Hvor mange rigtige han/hun havde - Hvilket spørgsmål han/hun havde svaret forkert på - Det rigtige svar på de spørgsmål han/hun havde svaret forkert på

9.3.1 USE CASE MODEL - EVALUERINGS SKEMAER

Use case diagram:



9.3.1.1 UC-ES1 Administrer Skabeloner

Navn:	UC-ES1 Administrer Skabeloner
Forfatter	Rasmus Reitz
Noter:	Mål: 1. At få oprettet og gemt en skabelon i systemet, til anvendelse ved oprettelse af nye skemaer 2. Brugeren vælger blandt de eksisterende skabeloner den ønskede og sletter denne, herefter er den fjernet fra systemet Primær aktør: System Administrator

Scenarier:

Type:	Formål:	Beskrivelse:
Succes	Slet skema	1. Aktøren åbner listen med eksisterende skabeloner 2. Aktøren udpeger den skabeloner der ønskes fjernet 3. Når skabelonen er markeret vælges slet 4. Den valgte skabelon slettes fra systemet
Succes	Tilføj Skema	1. Aktøren vælger at oprette en ny skabelon 2. Aktøren angiver navn og type på skabelonen 3. Der angives felter til skabelonen efter behov med tilhørende attributter 5. Den færdige skabelon gemmes i systemet
Alternativ	Udvidelser Slet skema	a. Hvis Aktøren ønsker at slette en skabelon, vælges objektet fra listen over eksisterende skabeloner, herefter følges proceduren som beskrevet. Jvf. forudsætningerne skal man være valideret som System administrator for at kunne slette en skabelon
Alternativ	Udvidelser tilføj skema	a. Hvis Aktøren på noget tidspunkt i scenariet vælger at afbryde oprettelsen af skabelonen bringes systemets tilstand til starttilstanden og ingen ny data er gemt i systemet 2a. Hvis Aktøren navngiver skabelonen identisk med et allerede eksisterende skema skal brugeren konfronteres herom, og navnet skal ændres.

Constraints:

Type:	Formål:	Beskrivelse:
Pre-condition	Brugeren er autoriseret	Brugeren er autoriseret til at måtte udføre handlingen
Pre-condition	Brugeren er valideret	Brugeren er blevet valideret som system administrator
Post-condition	skabelonen er slettet eller oprette	Den valgte skabelon er slettet eller oprettet i systemet

9.3.1.2 UC-ES2 Administrer Skemaer

Navn:	UC-ES2 Administrer Skemaer
Forfatter	Rasmus Reitz
Noter:	Mål: 1. At få oprettet og gemt et nyt skema i systemet. 2. At Slette et eksisterende skema i systemet. Primær aktør: SuperBruger

Scenarier:

Type:	**Formål:**	**Beskrivelse:**
Succes	Slet skema	1. Aktøren åbner listen med eksisterende skemaer 2. Aktøren udpeger det skema der ønskes fjernet 3. Når skemaet er markeret vælges slet skema 4. Det valgte skema slettes fra systemet
Succes	Tilføj Skema	1. En aktør ønsker en ny evaluering, og har derfor behov for et nyt skema, og vælger derfor at oprette et nyt skema. 2. Aktøren vælger hvilken type og skabelon der skal anvendes og navngiver det nye skema således at det er genkendeligt fremover. 3. Aktøren opretter det ønskede antal felter. For hver felt angives: - Felt navn - Felt Type - Felt Længde Der angives desuden i tilfælde af protokolskema - Vægt - min/max score 4. Når skemaet har det ønskede format gemmes dette i systemet.
Alternativ	Udvidelser	a. Hvis Aktøren ønsker at slette en skabelon, vælges objektet fra listen over eksisterende skabeloner, herefter følges proceduren som beskrevet. Jvf. forudsætningerne skal man være valideret som System administrator for at kunne slette en skabelon
Alternativ	Udvidelser tilføj skema	a. Hvis aktøren på noget tidspunkt i senariet vælger at afbryde oprettelsen af skemaet bringes systemets tilstand til starttilstanden og ingen ny data er gemt i systemet 2a. Hvis aktøren navngiver skemaet identisk med et allerede eksisterende skema skal brugeren konfronteres herom, og navnet skal ændres.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren er valideret som bruger i systemet
Pre-condition	Brugeren er autoriseret	Den pågældende bruger er autoriseret til at oprette nye skemaer i systemet
Post-condition	Operationen er fulført	1. Det ny oprettede skema gemmes i systemet til fremtidig brug. eller 2. Det valgte objekt er slettet fra systemet

9.3.1.3 UC-ES3 Foretag Evaluering

Navn:	UC-ES3 Foretag Evaluering
Forfatter	
Noter:	Mål: Det er målet at foretage en evaluering, af en medarbejder Primær aktør Leder

Scenarier:

Type:	**Formål:**	**Beskrivelse:**
Succes	Gennemfør evaluering	1. Aktøren printer en samlyt protokol fra systemet, samt et feedback skema. 2. Aktøren lytter til x antal samtaler ved siden af eller på afstand af medarbejderen 3. Aktøren scorer de enkelte faser i en samtale på en skala 1-7 jf. protokollen og noterer scoringerne på protokollen sammen med oplysninger om: - Medarbejderens navn og/eller bruger ID i systemet - Kaldstype ud fra prædefineret liste af muligheder for denne kundeprofil - Antal kald der er lyttet på - Dato for samlyt 4. Aktøren gennemfører en feedback samtale med medarbejderen, hvor observationer og scoringer gennemgås og diskuteres. Desuden udfyldes feedback skemaet. 5. Efter endt evaluering indtaster aktøren de to skemaer ind i systemet og de gemmes som en foretaget evaluering.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal have autorisation.	Brugeren skal være autoriseret til at foretage en evaluering
Post-condition	Evaluering er gemt	Den gennemførte evaluering er gemt i systemet

9.3.1.4 UC-ES4 Hent foretaget evaluering

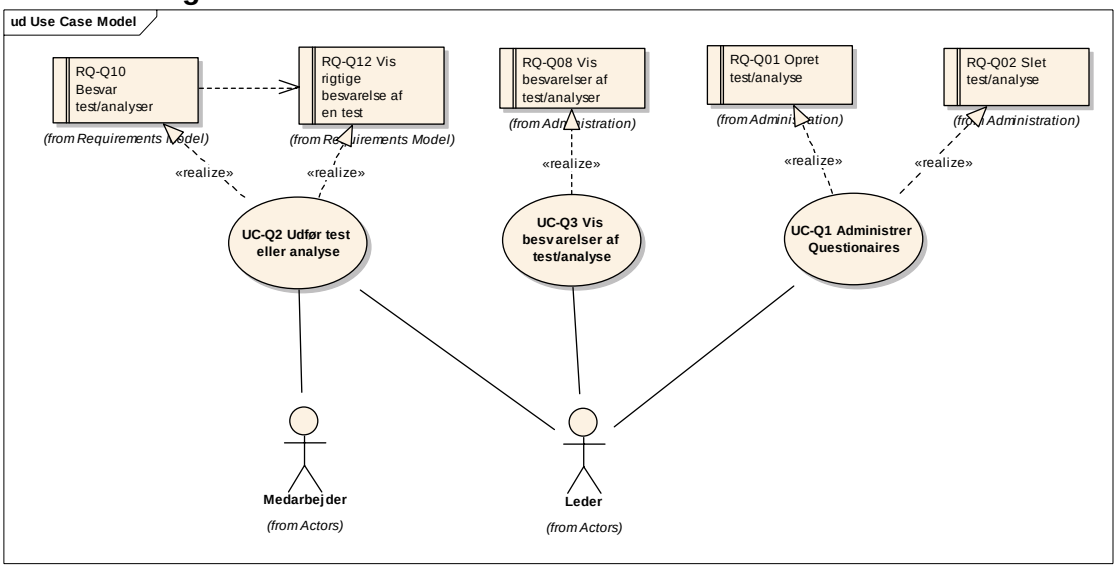
Navn:	UC-ES4 Hent foretaget evaluering	
Forfatter		
Noter:	Mål: Aktøren henter en tidligere foretaget evaluering. Primær aktør: Leder	
Scenarier:		
Type:	**Formål:**	**Beskrivelse:**
Succes	Hent evaluering	1. Aktøren vælger medarbejder for hvem der skal hentes gammel evaluering. 2. Aktøren vælger herefter fra en liste af foretagende evalueringer den ønskede evaluering 3. Den valgte evaluering vises herefter.
Constraints:		
Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren er valideret i systemet som minimum team leder
Pre-condition	Brugeren er autoriseret	Brugeren er autoriseret til at kunne hente gamle evalueringer.
Post-condition	Evalueringen bliver vist	Den gamle evaluering er hentet fra systemet og bliver vist.

9.3.1.5 UC-ES5 Foretag kalibrering

Navn:	UC-ES5 Foretag kalibrering	
Forfatter		
Noter:	Mål: En række aktører foretager en kalibrering og gemmer denne i systemet. Primær aktør: Leder	
Scenarier:		
Type:	**Formål:**	**Beskrivelse:**
Succes	Kalibrering	En gruppe ledere sætter sig sammen og foretage en kalibrering. Gruppen gennemlytter en række optagede samtaler. Lederne scorer individuelt de lyttede samtaler Gruppen diskuterer herefter resultaterne med henblik på at afstemme opfattelsen af god kvalitet
Constraints:		
Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren er valideret i systemet med rettigheder som minimum team leder
Pre-condition	Brugeren er autoriseret	Den validerede bruger er autoriseret til at oprette en kalibrering i systemet
Post-condition	Kalibreringen er gemt	Den registrerede kalibrering er gemt i systemet

9.3.2 USE CASE MODEL - QUESTIONAIRE

Use case diagram:



9.3.2.1 UC-Q1 Administrer Questionaires

Navn:	UC-Q1 Administrer Questionaires
Forfatter	
Noter:	Mål: At foretage administrative opgaver af tests og analyser. Primær aktør: Leder

Scenarier:

Type:	Formål:	Beskrivelse:
Succes	Oprettelse af analyse	Aktøren ønsker at oprette en analyse og skal angive følgende 1: Spørgsmål og tilhørende svarmuligheder. 2: Tidsfrist i klokkeslæt og dato for seneste besvarelse af analysen. 3: Hvilke brugere og/eller grupper der skal gennemføre testen. 4: Om besvarelser skal være anonyme eller ej. Når aktøren har angivet ovenstående, gemmes analysen i systemet og den udgives, således at analysen er tilgængelig for de i punkt 3 angivende brugere, således de kan besvare analysen.
Succes	Slet test/analyse	En aktør vælger en eksisterende test/analyse der ønskes slettes. Aktøren promtes for en bekræftelse af at denne test/analyse ønskes slettet. Hvis dette bekræftes slettes testen/analysen fra systemet
Alternativ	Udvidelser	Oprettelse af analyse: Hvis aktøren under oprettelse af en analyse fortryder oprettelsen, skal systemet returnere til tilstanden før oprettelsen blev påbegyndt. Evt. data der er gemt i systemet skal slettes.

Constraints:

Type:	Formål:	Beskrivelse:
Pre-condition	Brugeren er valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal have autorisation.	Brugeren skal være autoriseret til at oprette en analyse
Post-condition	Den ønskede operation er udført	

9.3.2.2 UC-Q2 Udfør test eller analyse

Navn:	UC-Q2 Udfør test eller analyse
Forfatter	
Noter:	Mål: At gennemføre en test eller analyse Primær aktør: Medarbejder

Scenarier:

Type:	**Formål:**	**Beskrivelse:**
Succes	Gennemfør analyse/test	1: Aktøren ønsker og besvarer en analyse. 2: Der svares på et spørgsmål ad gangen, indtil alle spørgsmål er besvaret. 3: Når alle spørgsmål er besvaret, skal aktøren bekræfte at besvarelsen gemmes i systemet. 4: Besvarelsen gemmes i systemet med information om hvornår besvarelsen er foretaget. 5: Hvis en aktør har afsluttet en test, skal de rigtige svar til spørgsmålene præsenteres. Følgende præsenteres for brugeren: - Hvor mange rigtige svar aktøren have, ud af hvor mange testen indeholdte - Hvilket spørgsmål aktøren havde svaret forkert på - De rigtige svar til de spørgsmål aktøren havde svaret forkert på
Succes	Udvidelser	2a: Aktøren kan gå tilbage til de tidligere spørgsmål og ændre i sin besvarelse. 2b: Aktøren kan afbryde besvarelsen, og vil blive præsenteret med muligheden for enten at: 1) Gemme besvarelsen hvor han/hun er nået til, og senere genoptage besvarelsen fra dette punkt. 2) Ignorere allerede besvarede sprøgsmål og starte forfra på besvarelsen senere. 3a: Hvis analysen ikke er angivet som anonym, informeres aktøren om, at det vil fremgå af besvarelsen hvem der har udfør den. 4a: Hvis analysen ikke er angivet som anonym gemmes ligeledes hvilken aktøre der har udført besvarelsen.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren skal valideres af systemet
Pre-condition	Brugeren er autoriseret	Brugeren skal autoriseres i systemet til at besvare den givne analyse
Pre-condition	Brugeren er i målgruppen for analysen	Den givne analyse er tildelt til besvarelse af brugeren, eller en gruppe brugeren er medlem af.
Pre-condition	Analysen/testen er gennemført	Den udgivne analyse/test er besvaret og besvarelsen er gemt i systemet

9.3.2.3 UC-Q3 Vis besvarelser af test/analyse

Navn:	UC-Q3 Vis besvarelser af test/analyse
Forfatter	Peter Thornton
Noter:	Mål: At vise besvarelserne til en test/analyse Primær aktør: Leder

Scenarier:

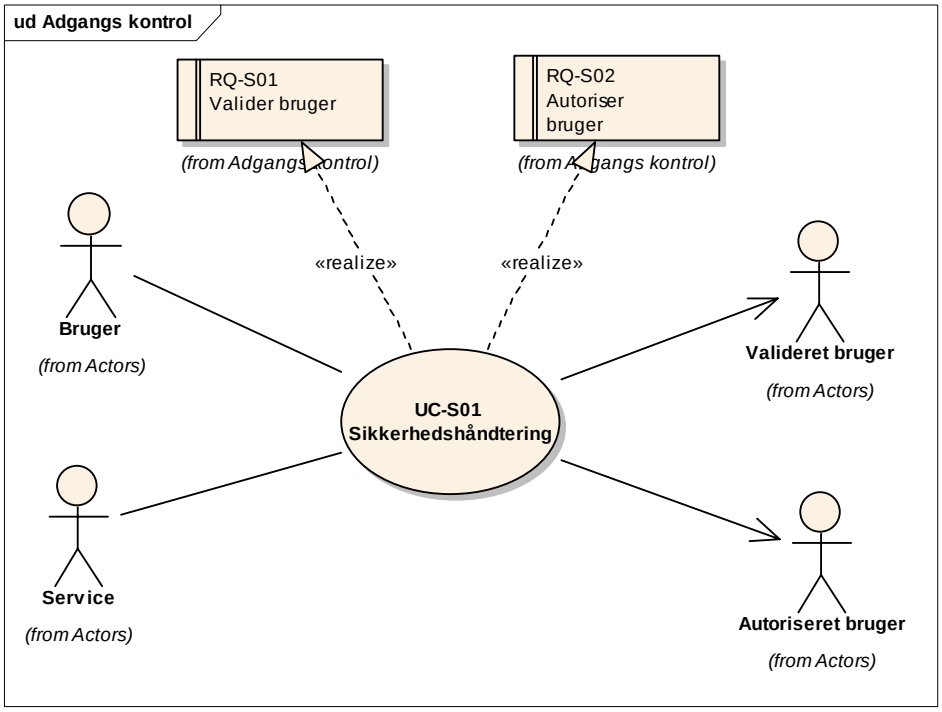
Type:	**Formål:**	**Beskrivelse:**
Alternativ	Anonyme besvarelser	Hvis testen/analysen er angivet som anonym, skal det ikke fremgå af præsentationen af besvarelserne hvordan de enkelte brugere har svaret. Er testen/analysen ikke angivet som anonym, skal det fremgå hvordan de enkelte brugere har svaret.
Succes	Vis besvarelser	En aktør vælger en eksisterende test/analyse og ønsker at se besvarelserne til denne. Besvarelserne præsenteres herefter for aktøren.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren skal være valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal være autoriseret	Brugeren skal være autoriseret til at se besvarelser af en test/analyse
Post-condition	Besvarelser er vist	De gemte besvarelser bliver vist på skærmen med en passende præsentation.

9.3.3.1 Adgangs kontrol - Use Case Model

Use case diagram:



9.3.3.1.1 UC-S01 Sikkerhedshåndtering

Navn:	UC-S01 Sikkerhedshåndtering
Forfatter	
Noter:	Mål: At kunne kontrollere om en brugere er valideret og autoriseret til at anvende en operation i en service. Primær aktør: Services Valideret bruger

Scenarier:

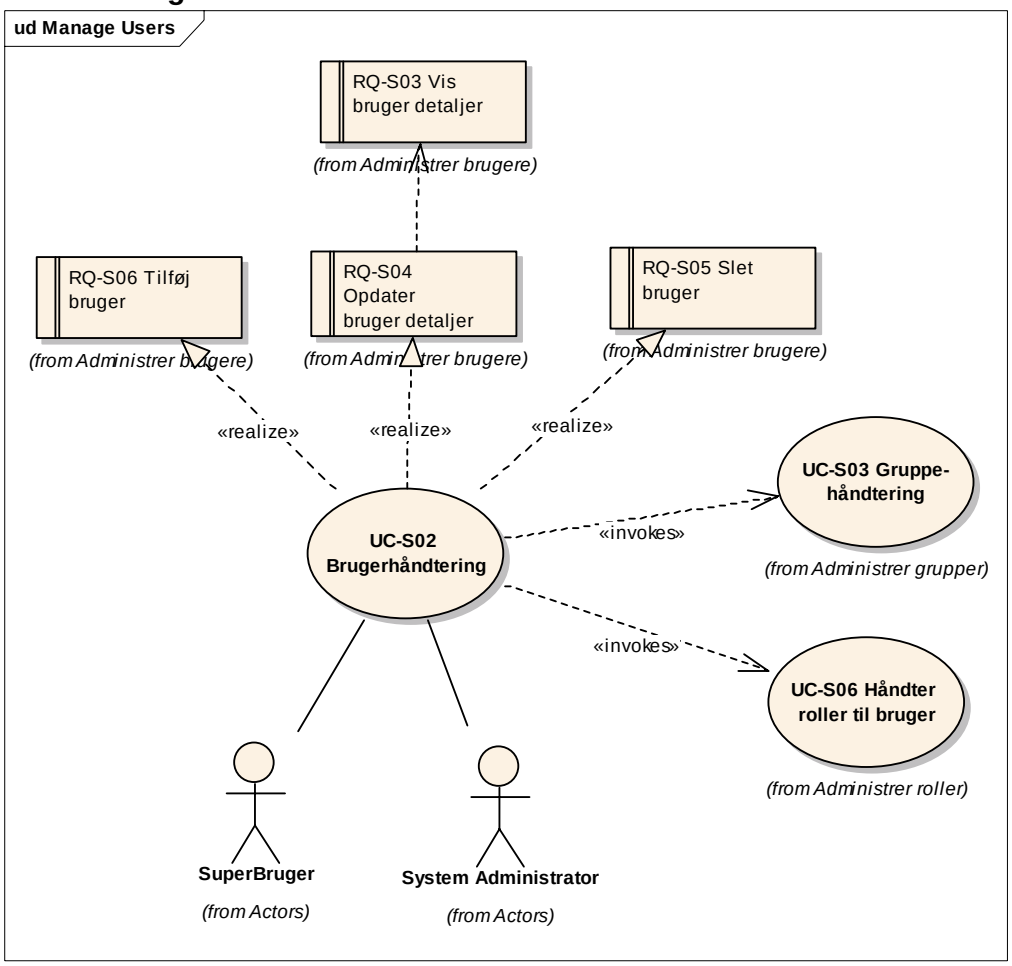
Type:	Formål:	Beskrivelse:
Succes	Autoriser bruger	1. En bruger logger ind på systemet med sit brugernavn og password 2. Brugeren brugernavn og password kontrolleres i systemet 3. Brugernavn og password er korrekt og brugeren er valideret startsiden vises 4. En bruger forespørger, med brugernavn og password, en service om adgang til en operation. 5. Servicen spørger herefter om brugeren med det angivende brugernavn og password må udføre operationen som servicen tilbyder. 6. Hvis brugeren har korrekte rettigheder tillades brug af operationen.
Alternativ	udvidelser	Hvis brugeren ikke indtaster det rigtige brugernavn og/eller password. bliver brugeren ikke logget ind i systemet og Hvis en bruger ikke har rettigheder til at anvende den pågældende operation, vil denne ikke kunne anvendes Alternativt kan brugerfladen kun tilbyde de operationer en valideret bruger har rettigheder til

Constraints:

Type:	Formål:	Beskrivelse:
Post-condition	Brugere er valideret og autoriseret	Brugere er valideret og autoriseret af systemet

9.3.3.2 Administrer brugere - Use Case Model

Use case diagram:



9.3.3.2.1 UC-S02 Brugerhåndtering

Navn:	UC-S02 Brugerhåndtering
Forfatter	
Noter:	Mål: At oprette, slette eller opdatere brugere i systemet. Primær aktør. Superbruger System Administrator

Scenarier:

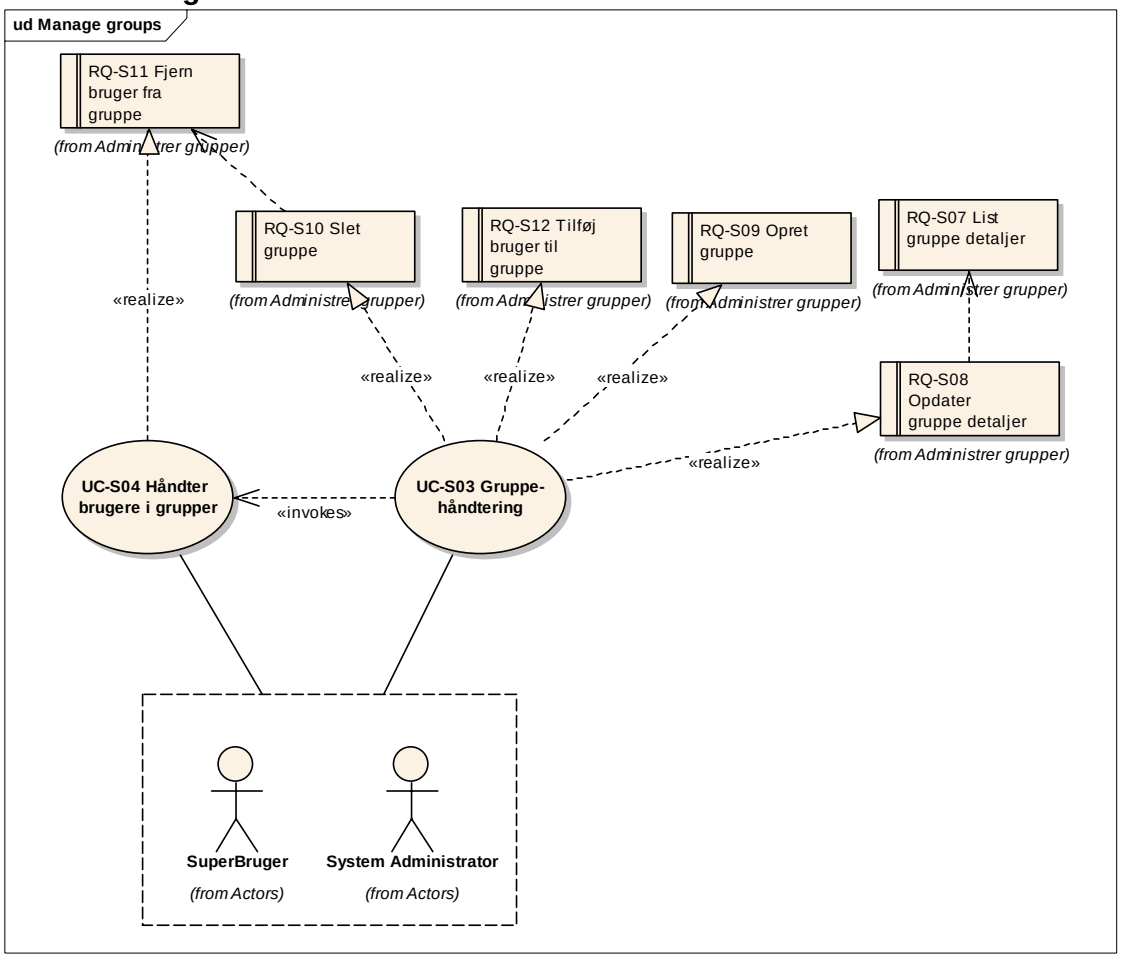
Type:	Formål:	Beskrivelse:
Succes	Opdater bruger	1. Aktøren ønsker at ændre nogle detaljer for en bruger. 2. Den pågældende bruger vælges blandt de eksisterende brugere. 3. Detaljerne for den valgte bruger ændres. 4. Brugeren gemmes i systemet igen med de nye detaljer.
Succes	Slet bruger	1. Aktøren ønsker at slette en bruger fra systemet som ikke længere er i brug. 2. Brugeren der ønskes slettet vælges blandt systemets eksisterende brugere. 3. den valgte bruger slettes fra systemet.
Succes	Tilføj bruger	1. Aktøren vælger at tilføje en ny bruger til systemet. 2. Til den nye bruger indtastes der information som angivet i RQ-S18. 3. Brugeren oprettes i systemet
Alternativ	udvidelser	Når information om den nye bruger er indtastet, kan aktøren vælge ligeledes at tilføje brugeren til eksisterende grupper eller roller.

Constraints:

Type:	Formål:	Beskrivelse:
Pre-condition	Aktøren er valideret	Brugeren er valideret i systemet
Pre-condition	Atøren autoriseret	Brugeren er autoriseret til at måtte tilføje brugere
Post-condition	Handlingen lykkedes	Handlingen med enten at slette, oprette eller opdatere en bruger er lykkedes.

9.3.3.3 Administrer grupper - Use Case Model

Use case diagram:



9.3.3.3.1 UC-S03 Gruppe-håndtering

Navn:	UC-S03 Gruppe-håndtering
Forfatter	
Noter:	Mål: At udføre handlinger vedrørende styring af grupper. Primær aktører: System Administrator Superbruger

Scenarier:

Type:	Formål:	Beskrivelse:
Succes	At opdatere en gruppes detaljer	En aktør ønsker at opdatere/rette en gruppes detaljer. Aktøren vælger gruppen der ønskes ændret. Gruppes nuværende detaljer listes jvf. RQ-S07. Aktøren foretager de ønskede rettelser og vælger og gemme disse. Herefter gemmes ændringerne i systemet.
Succes	At slette en gruppe fra systemet	En primær aktør ønsker at slette en gruppe fra systemet. Aktøren vælger den ønskede gruppe og angiver denne skal slettes. Indeholder gruppen brugere, gøres aktøren opmærksom på dette, og at alle brugere i gruppen miste deres tilhørsforhold til gruppen, og evt. roller de bliver tildelt fordi de er med i gruppen. Aktøren bekræfter at gruppen skal slettes, hvor efter alle brugere fjernes fra gruppen vha. UC-S08, og denne derefter slettes i systemet
Alternativ	At slette en tom gruppe fra systemet	En primær aktør ønsker at slette en gruppe fra systemet. Aktøren vælger den ønskede gruppe og angiver denne skal slettes. Aktøren bekræfter at gruppen skal slettes, og denne slettes herefter fra systemet
Succes	Oprettelse af gruppe	En primær aktør ønsker at oprette en gruppe i systemet. Aktøren vælger at oprette gruppen og angiver de ønskede værdier for gruppens detaljer, jvf. RQ-S09. Når brugeren har angivet værdierne for gruppes detaljer bekræfter han oprettelsen af gruppen, og denne oprettes i systemet.

Constraints:

Type:	Formål:	Beskrivelse:
Pre-condition	Brugeren er valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren er autoriseret	Brugeren skal være autoriseret til at oprette en gruppe
Post-condition	Handling lykkedes	Handlingen med enten at slette, oprette eller opdatere en gruppe er lykkedes.

9.3.3.3.2

9.3.3.3.3 UC-S04 Håndter brugere i grupper

Navn:	UC-S04 Håndter brugere i grupper
Forfatter	
Noter:	Mål: At håndtere en brugeres relationer til en gruppe Primær aktører: System Administrator Superbruger

Scenarier:

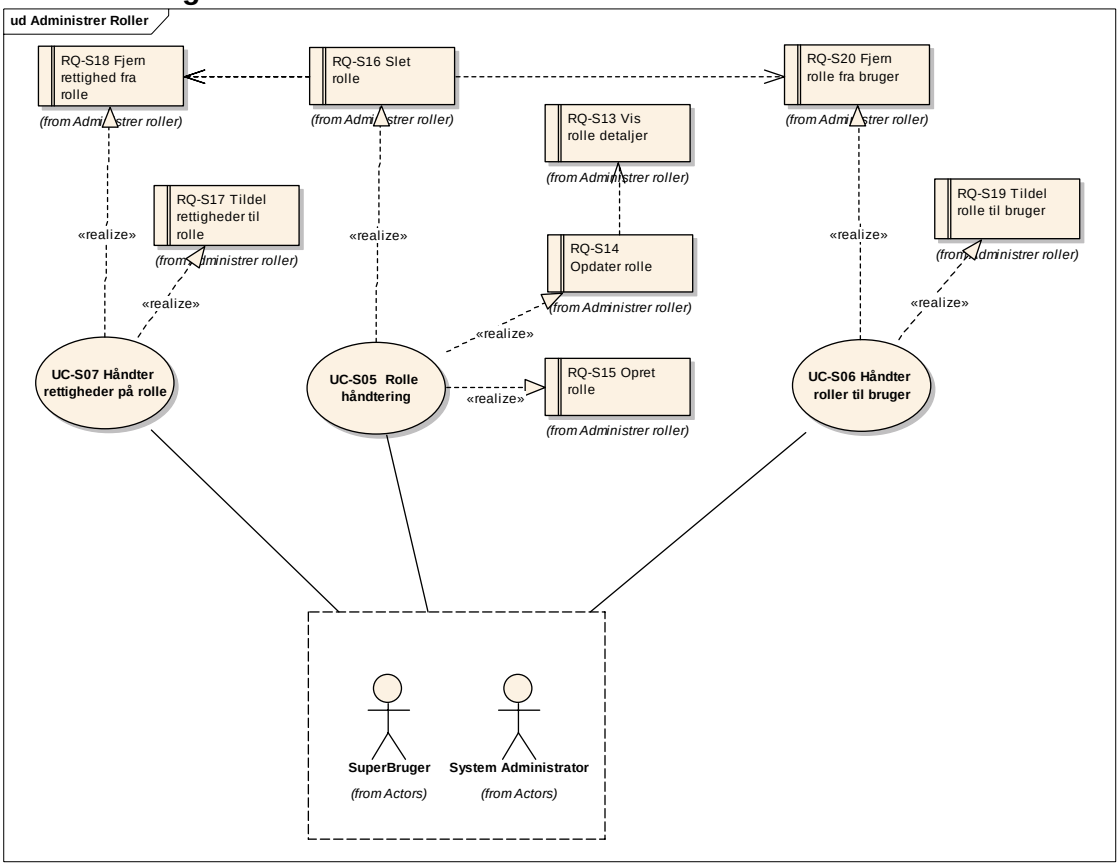
Type:	**Formål:**	**Beskrivelse:**
Succes	At fjerne en bruger fra en gruppe	En primær aktør ønsker at fjerne en bruger fra en gruppe. Aktøren vælger en ønskede bruger, og vælger på listen af grupper bruger en medlem af, hvor brugeren skal fjernes fra. Aktøren bekræfter at brugeren skal fjernes fra gruppen, og herefter gemmes ændringerne i systemet
Succes	Tilføjelse af bruger til gruppe	En primær aktør ønsker og tilføje en bruger til en gruppe. Aktørerne vælger først hvilken bruger der ønsket tilføjet. Herefter vælges hvilken gruppe denne bruger skal tilføjes. Aktøren bekræfter at den ønskede bruger skal tilføjes til den ønskede gruppe, og ændringen gemmes i systemet.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren skal være valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal være autoriseret	Brugeren skal være autoriseret til at tilføje en bruger til en gruppe
Post-condition	Brugeren tilhørsforhold er opdateret	Brugerens tilhørsforhold til en gruppe er opdateret

9.3.3.4 Administrer roller - Use Case Model

Use case diagram:



9.3.3.4.1 UC-S05 Rolle håndtering

Navn:	UC-S05 Rolle håndtering
Forfatter	
Noter:	Mål: At udføre handlinger vedrørende styring af roller Primær aktør System administrator Superbruger

Scenarier:

Type:	Formål:	Beskrivelse:
Succes	Opdater roller	1. En aktør vælger en rolle blandt de eksisterende roller. 2. Der rettes en eller flere detaljer på rollen. 3. Rollen gemmes i systemet igen.
Succes	Opret roller	Der er i systemet behov for en ny rolle. En aktør vælger at oprettet en ny rolle. Den nye rolle navngives og gemmes i systemet
Succes	Slet rolle	1. Der er i systemet ikke længere brug for en rolle. 2. Rollen udpeges i en liste over eksisterende roller 3. Rollen slettes fra systemet

Constraints:

Type:	Formål:	Beskrivelse:
Pre-condition	Brugeren er valideret	Brugeren der valideret i systemet
Pre-condition	Brugeren er autoriseret	Brugeren er autoriseret til at kunne oprette roller
Post-condition	Handling lykkedes	Handlingen med enten at slette, oprette eller opdatere en rolle er lykkedes.

9.3.3.4.2 UC-S06 Håndter roller til bruger

Navn:	UC-S06 Håndter roller til bruger
Forfatter	
Noter:	Mål: At tildele og fjerne roller fra brugere Primær aktør: SuperBruger System Administrator

Scenarier:

Type:	**Formål:**	**Beskrivelse:**
Succes	Fjern roller fra bruger	En aktør vælger en bruger i systemet som skal begrænses i sine rettigheder. På den valgte bruger vises en liste over de tildelte roller. Fra listen fjernes de roller der ikke længere skal være knyttet til den valgte bruger. Systemet opdateres og rollerne er fjernet fra brugeren.
Succes	Tildel roller til brugere	En aktør vælger en bruger i systemet. For den valgte bruger vælges der en eller flere roller. Rollerne tildeles den valgte bruger og det gemmes i systemet.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren er valideret i systemet
Pre-condition	Brugeren er autoriseret	Brugeren er autoriseret til at kunne tildele roller til brugere og grupper i systemet.
Post-condition	Brugeren er opdateret med de korrekte roller	Den pågældende bruger har fået frataget eller tildelt roller

9.3.3.4.3 UC-S07 Håndter rettigheder på rolle

Navn:	UC-S07 Håndter rettigheder på rolle
Forfatter	Rasmus Reitz
Noter:	Mål: At tildele og fjerne rettigheder fra en rolle der er i systemet. Primær aktør: Superbruger System Administrator

Scenarier:

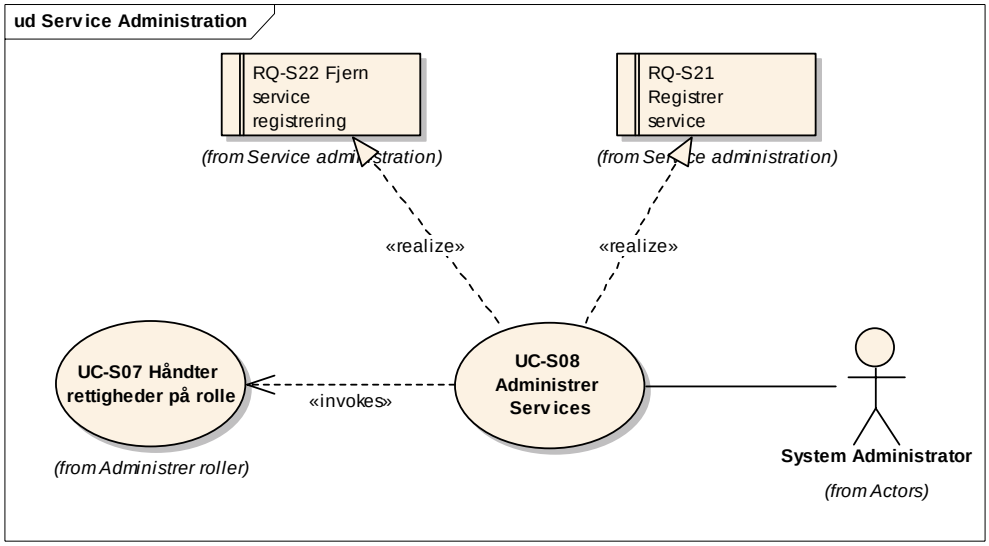
Type:	**Formål:**	**Beskrivelse:**
Succes	Fjern rettigheder fra rolle	1. En aktør vælger en rolle i systemet 2. På den valgte rolle fjernes de allerede tildelte rettigheder der ønskes fjernet. 3. Rollen gemmes i systemet igen.
Succes	Tildel rettigheder til roller	1. En aktør vælger en rolle i systemet. 2. For den valgte rolle tildeles en eller flere rettigheder til at udføre operationer som services registreret i systemet stiller til rådighed. 3. Rollen gemmes i systemet.

Constraints:

Type:	**Formål:**	**Beskrivelse:**
Pre-condition	Brugeren er valideret	Brugeren er valideret i systemet
Pre-condition	Brugeren er autoriseret	Brugeren er autoriseret til at kunne tildele rettigheder.
Post-condition	Rollen er opdateret	Rollen har fået frataget eller tildelt rettigheder

9.3.3.5 Service Administration - Use Case Model

Use case diagram:



9.3.3.5.1 UC-S08 Administrer Services

Navn:	UC-S08 Administrer Services
Forfatter	Peter Thornton
Noter:	Mål: Registrere eller fjerne en service i systemet. Primær Aktør: System Administrator

Scenarier:

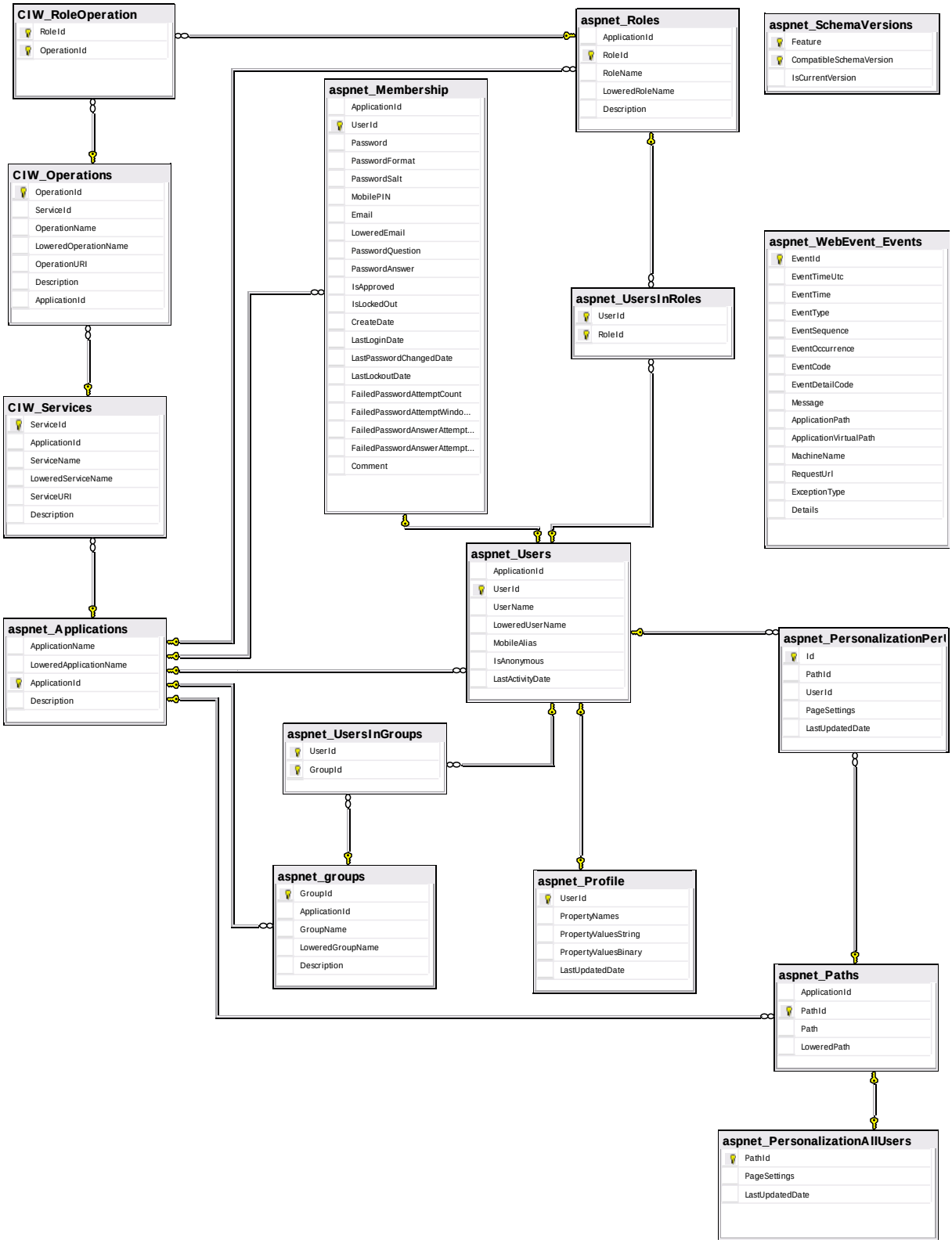
Type:	Formål:	Beskrivelse:
Succes	At fjerne en service registrering	En aktør ønsker at fjerne en service registrering fra systemet, således dennes funktionalitet ikke længere er tilgængelig. Aktøren bekræfter, at servicen skal fjernes fra systemet. Når servicen fjernes, skal alle rettigheder til operationer på denne service ligeledes fjernes fra roller hvor de måtte være tildelt.
Succes	At registrere en service	En aktør ønsker at registrere en service således dennes funktionalitet kan benyttes i systemet. Aktøren angiver et navn således servicen kan identificeres. Desuden angiver han hvilken forbindelses metode og adresse der skal benyttes ved kommunikation med servicen. Når der er oprettet forbindelse til services, forespørger systemet servicen efter en liste af operationer servicen tilbyder. Denne liste skal benyttes til at tildele roller rettigheder til at benytte operationerne fra servicen.

Constraints:

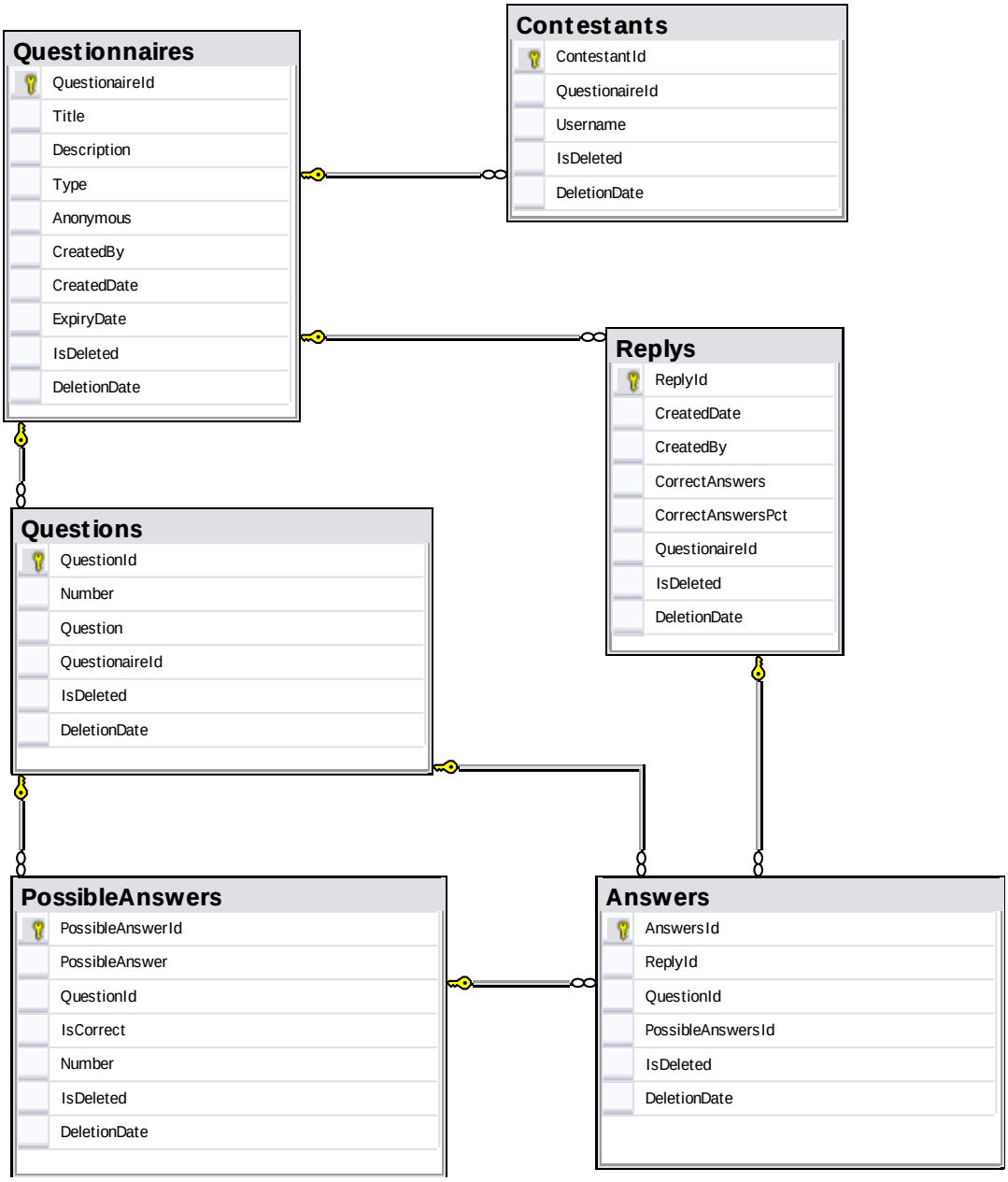
Type:	Formål:	Beskrivelse:
Pre-condition	Brugeren skal være valideret	Brugeren skal være valideret af systemet
Pre-condition	Brugeren skal være autoriseret	Brugeren skal være autoriseret til at registrere en service i systemet
Post-condition	Servicen er tilføjet eller fjernet	Den pågældende service er fjernet eller tilføjet fra systemet

9.4 BILAG 4 DATABASE DIAGRAMMER

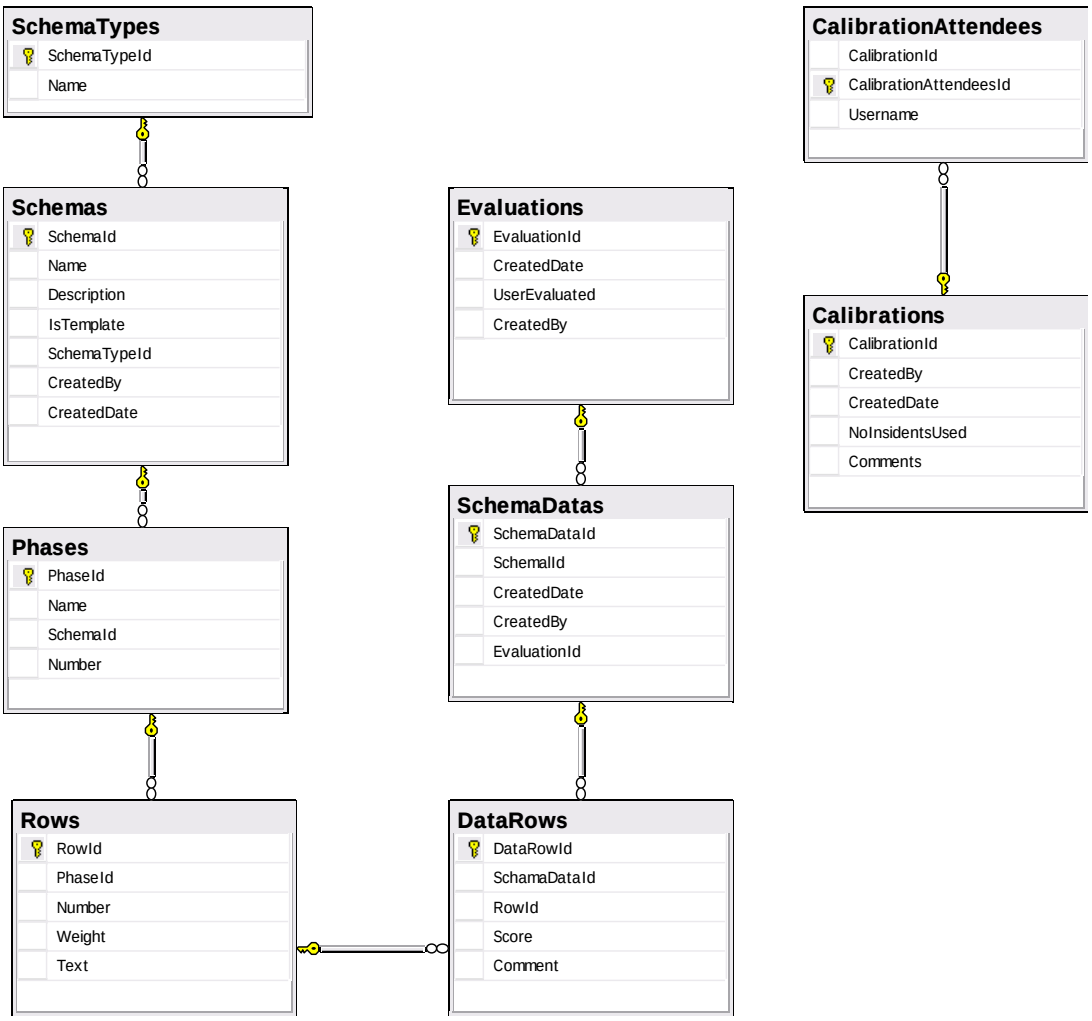
9.4.1 SYSTEMSERVICE DATABASE DIAGRAM



9.4.2 QUESTIONAIRE DATABASE DIAGRAM



9.4.3 EVALUERING DATABASE DIAGRAM



9.5 BILAF 5 SOAP

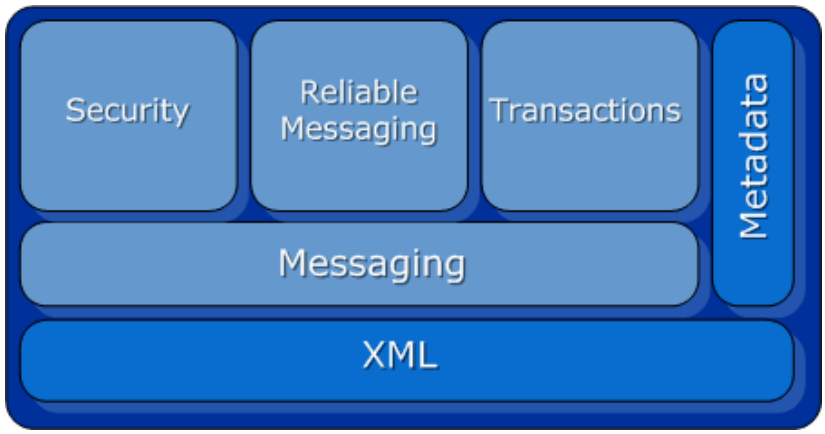
En SOAP besked består af en *envelope*, *body* og evt. en *header*. Strukturen i en SOAP besked ser ud som nedenstående

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  ..
  ..
</env:Header>
 <env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
  ..
  ..
 </env:Body>
</env:Envelope>
```

De fremhævede elementer, er de eneste SOAP specifikke elementer i beskeden. Indholdet af header og body er applikations afhængigt og det er op til designerne af den enkelte applikation at indsætte data i header og body elementerne. I eksemplet ovenstående er der indsat informationer om en rejse reservation.

9.6 BILAG 6: WS-* SPEFICIKATIONER

Microsoft har i samarbejde med bla. IBM og SAP defineret WS-* specifikationerne. WS-* specifikationerne bygger videre på WS-I Basic Profile specifikationerne og kan betegnes som 2. Generations web services. Udvidelserne omhandler bla. sikkerhed, troværdig besked udveksling, og transaktioner. Specifikationerne kan deles op i forskellige områder som vist i Figur.



Figur 47 – WS Specifikationsområder. © Microsoft Corporation

- **SECURITY**: Security området definere en række tilføjelser der omhandler validering, data integritet og data fortrolighed, således der opnås en sikker kommunikation mellem parterne.

- **MESSAGING**: Messaging indeholder bla. *WS-addressing* som gør SOAP uafhængig af, at den underliggende transport protokol skal indeholde adresserings oplysninger. Desuden indeholder det et optimeret transmissions format til SOAP beskeder.

- **METADATA**: Web services benytter WSDL til at beskrive en service og hvorledes denne service kan benyttes. *WS-Policy* der er en del af metadata specifikationerne giver ydermere mulighed for at beskrive f.eks. den foretrukne sikkerheds metode ved brug af en given service. Desuden indeholder det specifikationer for hvordan en klient kan forespørge om informationer om en service, som f.eks. dens WSDL og policies vha. *WS-metadataexchange* specifikationerne.

- **RELIABLE MESSAGING**: Som følge af risikoen for fejl i systemer, netværk mm. Definere WS-ReliableMessaging specifikationerne en protokol der muliggøre pålidelig beskedudveksling mellem parter på trods af system eller netværks fejl.

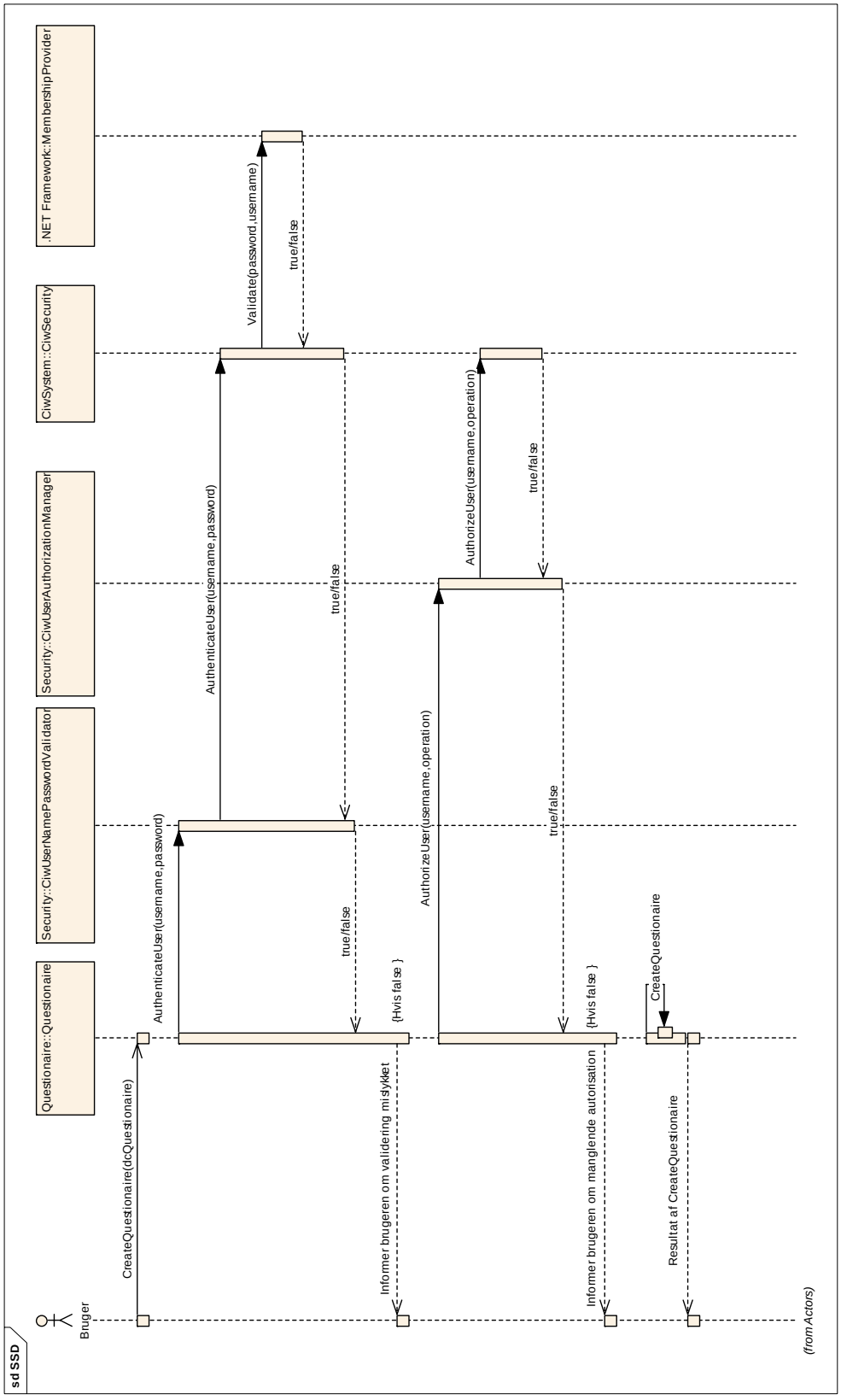
- **TRANSACTIONS**: Tilføjer understøttelse af transaktioner med SOAP besked udveksling, herunder bla. *two-phase commit* transaktioner.

WCF understøtter en række specifikationer inden for hvert af disse områder, således WCF kan tilbyde platforms uafhængige web services der er sikre, pålidelige med support for transaktioner mv.

9.7 BILAG 7 - BINDINGS TYPER

Binding	Interoperability	Security (Default)	Session (Default)	Transactions	Duplex
BasicHttpBinding	Basic Profile 1.1	(None), Transport, Message,	None, (None)	(None)	n/a
WSHttpBinding	WS	Transport, (Message), Mixed	(None), Transport, Reliable Session	(None), Yes	n/a
WSDualHttpBinding	WS	(Message)	(Reliable Session)	(None), Yes	Yes
WSFederationHttpBinding	WS-Federation	(Message)	(None), Reliable Session	(None), Yes	No
NetTcpBinding	.NET	(Transport), Message	Reliable Session, (Transport)	(None), Yes	Yes
NetNamedPipeBinding	.NET	(Transport)	None, (Transport)	(None), Yes	Yes
NetMsmqBinding	.NET	Message, (Transport), Both	(None)	(None), Yes	No
NetPeerTcpBinding	Peer	(Transport)	(None)	(None)	Yes
MsmqIntegrationBinding	MSMQ	(Transport)	(None)	(None), Yes	n/a

9.8 BILAG 8 SYSTEMSEKVENSDIAGRAM FOR SIKKERHEDSMEKANISME



9.9 BILAG 9 EKSEMPEL PÅ SELF-HOSTING

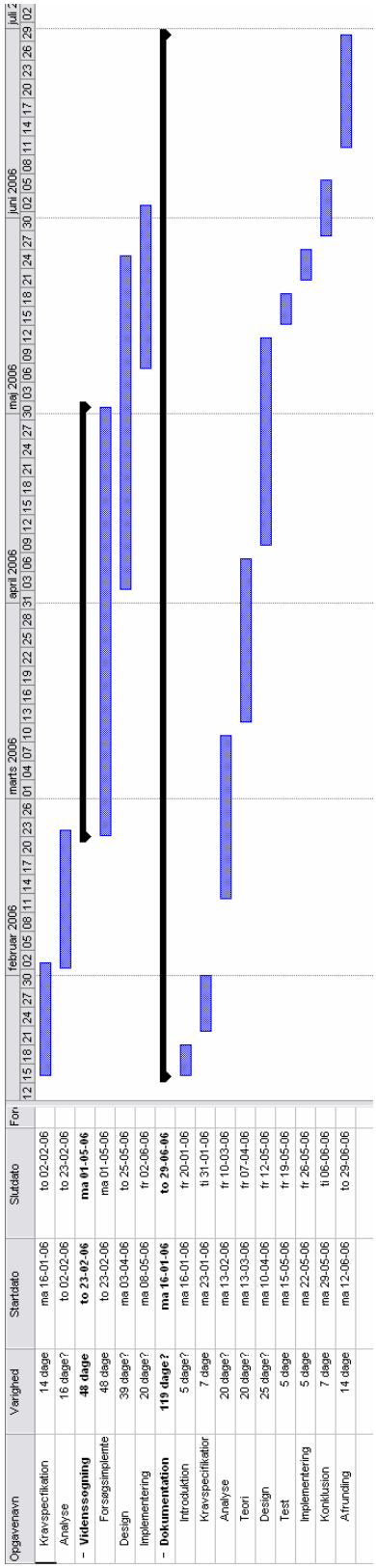
Som beskrevet i afsnit 4.5.5 er der under udviklingen benyttet self-hosted services i konsol applikationer. Dette er gjort, da det er nemmere og starte/stoppe services, og det desuden giver muligheden for at udskrive debug information direkte til konsollen.

```
static void Main(string[] args)
{

    try
    {
        // Get base address from appsettings in configuration
        Uri questbaseAddress = new Uri(ConfigurationManager.AppSettings["questbaseAddress"]);

        // Create a ServiceHost for the Questionaire type with the baseaddress.
        Using (ServiceHost serviceHost = new
            ServiceHost(typeof(Ciw.CiwService.Questionaire.Questionaire), questbaseAddress))
        {
            // Open/Start the ServiceHost
            serviceHost.Open();

            //Write info
            Console.WriteLine("The service  :" + serviceHost.Description.Name);
            Console.WriteLine("In namespace :" + serviceHost.Description.Namespace + " is ready.");
            Console.WriteLine("Running in the following account: {0}",
            WindowsIdentity.GetCurrent().Name);
            Console.WriteLine("Running at: " + questbaseAddress + " with the following endpoints:");

            //Iterate thru endpoints
            foreach (System.ServiceModel.Description.ServiceEndpoint point in
                serviceHost.Description.Endpoints)
            {
                Console.WriteLine("Contract: " + point.Contract.Name + " Address: " +
                point.Address.Uri);
            }

            Console.WriteLine("Press <ENTER> to terminate service.");
            Console.WriteLine();

            //Wait for keypress
            Console.ReadLine();

            // Stop/close the service
            serviceHost.Close();
        }
    }
    catch (Exception exp)
    {
        //Catch any exception and output it to the console
        Console.WriteLine(exp.Message);
        Console.ReadLine();
    }
}
```

I eksemplet ses det hvorledes self-hosting er implemeteret. Først hentes baseadressen fra konfigurationen, der er lagt her for at gøre det simpelt og skifte baseadressen for servicen. Herefter oprettes en instans af *ServiceHost* klassen. For at oprette denne angives der hvilken service der skal hostes. Bemærk at der angives den klasse hvor servicen er implemeteret dvs. "Questionaire" og ikke servicekontraktens klasse "IQuestionaire."

Her efter startes servicen og der udskrives en række informationer om den. Alle exceptions fanges og exception beskeden udskrives til konsollen.

9.10 BILAG 10 TIDSPLAN



Opgavenavn	Varighed	Startdato	Slutdato	For
Kravspecifikation	14 dage	ma 16-01-06	to 02-02-06	
Analyse	16 dage?	to 02-02-06	to 23-02-06	
- Videnssøgning	**48 dage**	**to 23-02-06**	**ma 01-05-06**	
Forsøgsimplemente	48 dage	to 23-02-06	ma 01-05-06	
Design	39 dage?	ma 03-04-06	to 25-05-06	
Implementering	20 dage?	ma 08-05-06	fr 02-06-06	
- Dokumentation	**119 dage?**	**ma 16-01-06**	**to 29-06-06**	
Introduktion	5 dage?	ma 16-01-06	fr 20-01-06	
Kravspecifikation	7 dage	ma 23-01-06	ti 31-01-06	
Analyse	20 dage?	ma 13-02-06	fr 10-03-06	
Teori	20 dage?	ma 13-03-06	fr 07-04-06	
Design	25 dage?	ma 10-04-06	fr 12-05-06	
Test	5 dage	ma 15-05-06	fr 19-05-06	
Implementering	5 dage	ma 22-05-06	fr 26-05-06	
Konklusion	7 dage	ma 29-05-06	ti 06-06-06	
Afrunding	14 dage	ma 12-06-06	to 29-06-06	