



Indholdsfortegnelse

1. Indledning	3
1.1 Forord.....	3
1.2 Indledning	4
1.3 Opgaveformulering.....	5
2. Analyse	6
2.1 Indledning til kravmodel.....	7
2.2 Funktions-forslag.....	8
2.3 Funktions-afgrænsning	10
2.4 UseCase-model	13
2.5 Domæne Model	16
2.6 Mock-ups.....	18
2.7 Konklusion på kravmodel.....	20
2.8 Indledning til analysemodel	21
2.9 ICE-model.....	21
2.10 Konklusion på analysemodel	23
3. Design	24
3.1 Teknologi-valg	25
3.2 Designovervejelser	27
3.3 Tilrettet Domæne Model	32
3.4 Database-beskrivelse	34
3.5 MAOV's egne tabeller	39
3.6 Sekvens-diagrammer.....	42
3.7 Konklusion	45
4. Implementering.....	46
4.1 Klassediagram	47
4.2 Manøvrering i koden.....	49
4.3 SQL-kommandoer.....	50
4.4 Hvordan gemmes valgte kunder og... ..	55
4.5 Input-argumenter	57
4.6 Hvordan ændres systemet	59
4.7 Forslag til forbedringer.....	61
4.8 Konklusion	62
5. Test	63
5.1 Teknologi-test	64
5.2 Design-test	64
5.3 Whitebox-test	65
5.4 Blackbox-test	65
5.5 UseCase-test.....	66
5.6 Bruger-test	67
5.7 Konklusion	68
6. Afslutning	69
6.1 Konklusion på systemet.....	70



7. Proces.....	71
7.1 Opgavens karakter.....	72
7.2 Kritiske succesfaktorer.....	73
7.3 Ekstern projektplan.....	74
7.4 Gennemgang af projektplan.....	76
7.5 Anvendte værktøjer.....	77
7.6 Kvalitetsstyring.....	78
7.7 Dagbog.....	79
7.8 Konklusion.....	81
8. Bilag	82
8.1 Oracle Danmarks's afdelinger.....	83
8.2 Muligheden for ændringer i virksomheden	85
8.3 Oracle's MAOV-prototype.....	87
8.4 Generelt om OOSE.....	88
8.5 Round-trip	91
8.6 Design-pattern: Layers	92
8.7 WAP-signalets rute	94
8.8 Portal-to-Go – Hvad er det?.....	95
8.9 Portal-to-Go's XML-standard.....	97
8.10 Det endelige system (screenshots).....	99
8.11 Oprettelse af brugere til MAOV-systemet ...	102
8.12 Forslag til eksterne funktioner	103
8.13 Møde-referater.....	104
8.14 Bruger-interviews	112
8.15 Litteraturliste.....	120
8.16 Greetings.....	121



1.1 Forord

Denne rapport beskriver udviklingen af projekt MAOV, og er udviklet for Oracle Danmark, af et hold studerende fra Lyngby Uddannelses Center (LUC), som hovedopgave til deres datamatiker-uddannelse.

Hovedopgaves formål er at bevise at de studerende har opnået de kvalifikationer der er krævet for en færdiguddannet datamatiker. Dette skal resultere i en rapport der beskriver et projekt-forløb, som bygger på udvalgte dele af den tilegnede viden fra uddannelsen. Der er meget brede grænser for et sådant projekt, og det er derfor op til de studerende at finde et projekt der passer med deres interesser. Det vigtige er at hovedopgaven beviser evnen til at udføre et større projekt og inddrage de relevante teorier og værktøjer i løsningen.

Kontakten til Oracle Danmark blev skabt ved lidt af et tilfælde. Vi var blevet fortalt at det var muligt for studerende at lave hovedopgave der, så det var det første firma vi kontaktede. De var meget interesserede da vi fortalte at vi gerne ville arbejde med teknologierne WAP og XML. De havde nemlig et internt projekt der netop omhandlede disse teknologier, men som der ikke var ressourcer til at udvikle.

Projektet bygger på ideen til et system som et par af Oracle's medarbejdere har gået og puslet med. Det var derfor naturligt at de blev vores kontakt-personer under udførelsen af det komplette udviklingsforløb. De havde, før vi kom til, lavet en lille fungerende prototype af deres ide, men den var kun designet efter deres egne behov, og var på ingen måde dokumenteret. Der var derfor vores opgave at indsamle reelle brugerbehov, implementere en mere fuldkommen version af systemet og få det hele dokumenteret.

Projektet er udarbejdet i perioden 14.08.2000 til 09.11.2000, under vejledning af:

- Jørgen Karrebæk (LUC)
- Asger Jensen (Oracle DK)
- Søren Hebsgaard (Oracle DK)

Projekt MAOV's udviklerteam bestod af: Martin Atke Bentsen og Thomas T. Pedersen.

Martin Atke Bentsen

Thomas T. Pedersen

Ophavsrettigheder til Projekt MAOV forbeholdes projektets udviklerteam. Rapporten må dog frit citeres med angivelse af kilde.



1.2 Indledning

MAOV står for, Mobil Adgang til Online Virksomhedsoplysninger, og går ud på at give Oracle's medarbejdere nem og simpel adgang til oplysninger når de er ude hos en kunde, i bilen eller hjemme. I første omgang via en WAP-telefon, men systemet er også designet til at kunne udnytte fremtidige håndholdte enheder der kan kommunikere trådløst med internettet.

Systemet er udviklet op imod Portal-to-Go (herefter PtG), som er Oracle's portal-produkt til mobile enheder. Det giver den fordel at systemet kan benyttes på flere forskellige enheder, uden at være udviklet specifikt til hver enkelt. PtG kræver at MAOV-systemet leverer XML-sider, som PtG så kan formattere til de mange forskellige enheder. Der er en generel beskrivelse af PtG i bilag 8.8 side 95.

Projektets hovedfokus har været at få udviklet et færdigt brugbart system, og der har derfor været lagt stor vægt på implementeringsfasen. Oracle Danmark forventede at få et færdigt fungerende produkt, ved afslutning af projektet.

1.2.1 Målgrupper

Rapporten er delt op i kapitler efter de forskellige faser i udviklingsforløbet. Disse kapitler henvender sig til forskellige målgrupper. De to primære målgrupper er: "de studerendes uddannelsessted" og "udviklere hos Oracle DK", som hverisær kan få forskelligt ud af de forskellige kapitler. Her er derfor en generel beskrivelse af hvad de enkelte kapitler omhandler, og hvorfor man måtte ønske at læse dem.

1. **Indledning** - Dette kapitel giver et overblik over projektet og henvender sig til samtlige læsere.
2. **Analyse** - Beskriver hvilke krav systemet skal opfylde. Det kan læses som introduktion til designet eller som udgangspunkt for et fremtidigt tilsvarende system.
3. **Design** - Indeholder de overvejelser der lægger til grund for systemets design. Det kan læses hvis man ønsker indsigt i hvordan systemet er designet.
4. **Implementering** - Forklarer selve implementeringen af systemet, så et eventuelt senere udviklerteam kan fortsætte udviklingen.
5. **Test** - Beskriver de tests der er udført på det implementerede system, så man kan se om det lever op til forventningerne.
6. **Afslutning** - En afrundning af udviklingen, der opsummerer de opnåede resultater, så læseren kan få en konklusion på produktudviklingen
7. **Proces** - Dokumenterer processen under udarbejdelsen af projektet. Henvender sig til læsere der ønsker at få et indblik i hvordan projektet er styret og forløbet.
8. **Bilag** - Heri er de mere teoretiske beskrivelser af forskellige emner vedrørende projektet. Der bliver henvist til dem, relevante steder i rapporten.

1.2.2 Resultatet

MAOV-systemet er blevet færdigudviklet og klar til brug efter dette projekts afslutning. Det er udviklet ud fra brugernes behov, og opfylder de af dem, der var mulige at implementere.

1.2.3 Litteratur-henvisninger

Projektet er primært udarbejdet ud fra egne erfaringer, og fra datamatiker-uddannelsens 4 første semestre. De steder der er anvendt litteratur direkte, er det anvist med et lille hævet nummer. Eksempelvis ⁵⁾, som henviser til værk nummer fem i litteraturlisten. Litteraturlisten er vedlagt som bilag 8.15 side 120.



1.3 Opgaveformulering

1.3.1 Indledning

Oracle har fået et behov for at give mobil adgang til firmaets centrale oplysninger. Det kunne eksempelvis være en sælger der hurtigt vil skabe sig et overblik over hvilke produkter og licenser en kunde allerede har, for bedre at kunne rådgive ham.

Der vil være forskellige grupper af brugere til et sådant mobilt system, der hver har sine individuelle behov. Disse grupper er: sælgere, konsulenter, partnere, supportere og kunder. Det er vigtigt at grupperne ikke kan få adgang til hinandens oplysninger, da disse kan være fortrolige.

Hvilke oplysninger de forskellige grupper skal have til rådighed, vil ikke blive fastlagt før deres behov er blevet grundigt analyseret.

De oplysningerne der skal være tilgængelige mobilt, ligger spredt ud på flere databaser. Det bliver vores opgave at finde og samle disse oplysninger og præsentere dem overskueligt på den mobile enhed.

Som teknologien er nu vil den mobile enhed sandsynligvis være en WAP-telefon. Den nuværende WAP-standard vil dog snart blive afløst, så systemet skal nemt kunne tilpasses nye standarder for bærbare enheder.

Det er vigtigt for et firma som Oracle selv at benytte, og være på forkant med de nyeste teknologier. Dette giver et professionelt indtryk af virksomheden og vil gøre kunderne mere trygge ved køb.

1.3.2 Problemformulering

Give mobil og sikker adgang til centrale oplysninger, for sælgere, konsulenter, partnere, supportere og kunder.

1.3.3 Afgrænsning

Vi vil lave et fuldt udvikling-forløb, med analyse, design, implementering og test. Vi vil sætte det primære krav, at vi får implementeret et funktionelt system, der til eksamen kan fremvises. Dette er den første afgrænsning. Der foretages yderligere afgrænsning i afsnittene:

- 2.1.1 Valg af brugere/interviewpersoner (side 7)
- 2.3 Funktionsafgrænsning (side 10)
- 2.4.1 UseCase-model - Aktør: Bruger (side 13)
- 3.1 Teknologi-valg (side 25)
- 3.3 Tilrettet Domæne Model (side 32)
- 3.4.6 Databasebeskrivelse - afrundning (side 38)



2. Analyse

2.0.1 Indledning

I dette kapitel beskrives analyse-fasen, med dens to modeller: kravmodel og analysemodel. Kapitlet kan læses af enhver, og kræver ingen særlig teknisk indsigt. Det vil dog være en fordel at kende OOSE systemudviklingsmetoden, som er beskrevet generelt i bilag 8.4 side 88.

Analyse-fasens formål er at analysere brugernes behov som skal danne grundlag for systemet. Der bliver lagt vægt på at brugerne er med til at definere det endelige systems funktionalitet, da det er dem der skal bruge det i sidste ende.



2.1 Indledning til kravmodel

Kravmodellen definerer brugernes krav til systemet, som skal danne grundlag for det videre udviklingsforløb. Alle krav er fremkommet ved interviews og resultatet af disse interviews blev 12 forskellige funktioner. Efter en afgrænsning endte det med 8 funktioner, som der var grundlag for at arbejde videre med.

Usecase-model og Domæne Model er udarbejdet på baggrund af de 8 tilbageværende funktioner, dog er 3 af de afgrænsede funktioner afbildet på modellerne (stiplede linjer), da de bygger på reelle behov. Problemet med dem er, at det simpelthen ikke kan lade sig gøre at implementere dem rent teknisk og organisatorisk her i Oracle Danmark.

Der er ikke lavet mock-ups for alle funktioner, men kun for en enkelt funktion, hvor hele scenariet gennemgås. Dette er valgt da der reelt ikke er meget variation i skærbillederne til de forskellige funktioner, alt bliver bygget op omkring en menustruktur hvor så få tastetryk som muligt er i højsædet. Endvidere er et navigationsdiagram udarbejdet, for at give et overblik over hvordan man bevæger sig rundt i menustrukturen.

2.1.1 Valg af brugere / interviewpersoner

Da der fra projektets start kun eksisterede en opgavebeskrivelse, og ingen egentlig kravspecifikation, var det nødvendigt at interviewe de forskellige grupper af brugere for at få kravene til systemet defineret korrekt. Resumé af de enkelte interviews er vedlagt som bilag 8.14 side 112.

De forskellige grupper af brugere blev nøje udvalgt i samarbejde med Søren Hebsgaard (se mødereferat af 21.08.2000 bilag 8.13 side 104), så de repræsenterede et bredt udsnit af de enkelte afdelinger i Oracle Danmark. Grupperne kunder og partnere blev udeladt, hvilket vil sige at det kun var brugere internt i organisationen der skulle interviewes. De eksterne grupper blev valgt fra, fordi det viste sig at Oracle ikke tillader at eksterne personer får adgang til Oracle's interne net, så allerede her blev den første afgrænsning af opgaven foretaget. Det har dog ikke den store betydning for projektets omfang, da der stadigvæk er nok arbejde med de resterende grupper. Det betyder dog at der ikke længere er nogen fortrolige oplysninger, der skal holdes adskilt imellem de forskellige grupper af brugere, da der ikke er nogen hemmeligheder internt i organisationen.

I det følgende afsnit er alle de foreslåede funktioner beskrevet, og i det næste er hver enkelt funktion analyseret teknisk og organisatorisk, for at finde ud af hvilke der kan lade sig gøre at implementere.



2.2 Funktions-forslag

Herunder er en komplet liste over alle de funktioner brugerne har udtrykt ønske om. Det viste sig at det ikke var alle funktioner Oracle tillod implementeret, grundet tekniske og organisatoriske hindringer. Funktions-afgrænsningen er beskrevet i [næste afsnit](#).

- Kalender for samtlige medarbejdere
- Telefonliste over medarbejdere
- Time-registrering
- Marketings-oplysninger
- Adgang til sine e-mails
- Oplysninger om software-licenser
- Prisliste (produkter og konsulent-ydelser)
- Kontaktliste over kunder/partnere
- Oplysninger om aktuelle sager (TAR)
- Adgang til et udsnit af "Sales Online" systemet
- Har kunden betalt for tidligere køb
- Tildele action-points til kolleger

2.2.1 Kalender for samtlige medarbejdere

Det skal være muligt at kunne se sin egen, og ikke mindst andre medarbejders kalender. Så vil det være muligt at se om en given medarbejder er ledig på et givent tidspunkt, og man kan booke et møde.

Eksempel: En konsulent er til møde med en kunde. Det viser sig at konsulenten ikke selv kan løse opgaven uden at inddrage en specialist. Konsulenten bliver derfor nød til at aftale et nyt møde, hvor alle tre kan deltage. Dette kræver at han har adgang til specialistens kalender. I dag foregår dette ved at konsulenten skal ringe til specialisten, som muligvis kan være optaget eller ikke til at få fat på.

2.2.2 Telefonliste over medarbejdere

En simpel telefon-bog over samtlige af Oracle Danmark's medarbejdere, så man let kan finde de personer man ikke plejer at arbejde med. Det kunne kombineres med at man kan søge folk ud fra deres faglige ekspertise, så man kan finde specialisterne. Som det er nu, har de fleste medarbejdere en udprintet liste over samtlige telefonnumre.

2.2.3 Time-registrering

Time-registreringen skal indrapporteres hver mandag over et web-baseret system. Dette system er lidt klodset opbygget og tungt at bruge. Hvis man nu havde mulighed for at opdatere sit timeforbrug løbende nemt og enkelt, så man ikke først skal skrive timerne ned på papir eller gå og huske på dem, for så at indrapportere det hele samlet sidst på dagen/ugen.

2.2.4 Marketings-oplysninger

En sælger skal vide om en kunde er tilmeldt diverse seminarer og mailinglister, så han kan henvise til dem, når han er ude til møde med kunden. Det virker mere progressivt når sælgeren er orienteret, fremfor at skulle spørge kunden om sådanne ting. På den anden side, hvis kunden spørger under mødet, kan sælgeren hurtigt undersøge det, og eventuelt tilmelde kunden. Det skal desuden være muligt at se hvilke seminarer/kurser kunden har været på, og kunne se om der er fremtidige kurser af interesse.



2.2.5 Adgang til sine e-mails

Det skal være muligt for de enkelte medarbejder at kunne checke deres mail-box uanset hvor de er. Det er ikke ønsket at kunne besvare e-mails på mobiltelefonen, da det ikke er hensigtsmæssigt med mobiltelefonens indtastningsmuligheder, som de er idag. Det vigtige er at kunne se om man har modtaget den vigtige meddelelse man venter på.

2.2.6 Oplysninger om software-licenser

Man skal nemt kunne få en oversigt over hvad en kunde har af licenser, for at kunne foreslå nye løsninger der bedst passer med den eksisterende produkt-portefølge. Det er kun få oplysninger der er nødvendige: Hvilke produkter, hvor mange licenser og hvilken platform de kører på.

2.2.7 Prislister (produkter og konsulent-ydelser)

Prislisten er ikke altid lige tilgængelig i en opdateret version. Når den er printet ud på papir, risikerer man at det er en forældet version man har med sig. Det ville derfor være meget brugbart hvis den aktuelle prisliste altid var online.

2.2.8 Kontaktliste over kunder/partnere

De fleste medarbejdere samler visit-kort på deres kontaktpersoner. De oplysninger bliver dog samtidigt gemt i Oracle's kundedatabase. Hvis der var mobil adgang til kontakt-oplysningerne, kunne den enkelte medarbejder slippe for at samle samtlige visitkort, og dermed risikere at deres samling ikke er komplet.

2.2.9 Oplysninger om aktuelle sager (TAR)

Supporterne opretter en såkaldt "TAR" for hver kunde-henvendelse. Det kan eksempelvis være et problem med en database. I den TAR opdateres så løbende hvordan sagen forløber, og hvor højt den er prioriteret, samt om det er kunden der venter på et svar fra supporteren, eller om det er supporteren der venter på yderligere oplysninger fra kunden. Disse oplysninger er det vigtigt for en sælger at være informeret om, inden han skal til et kundemøde.

2.2.10 Adgang til et udsnit af "Sales Online" systemet

I Sales Online står bl.a oplysninger om hvilke muligheder der er for salg hos de forskellige kunder og hvem der er deres kontaktpersoner. Sådant et system sælger Oracle også, og nogle kunder forespørger WAP-adgang til det. Der er kun een konkurrent der tilbyder WAP-adgang til noget lignende, og de har kun et log-op til Finland at vise frem. Citat Michael Skaarup Christiansen: "Det ville være r..blæret at kunne vise en fungerende version, der kører i DK og hvor man eksempelvis kan slå kunden selv op!!!"

2.2.11 Har kunden betalt for tidligere køb

Som sidebemærkning er det blevet foreslået at man kun vil sælge til kunder der har betalt for tidligere køb. Mange internetbutikker laver ikke denne kontrol, og man kan derfor blive ved med at bestille uden at betale for noget som helst. Denne kontrol-funktion skal ikke laves som en separat funktion, men vil kunne integreres i andre funktioner.



2.2.12 Tildele action-points til kolleger

Efter et kundemøde er der en række action-points der skal udføres. Nogle af dem skal udføres af andre, og det ville være rart hvis man kunne tildele dem direkte til de respektive kolleger på stedet, ved at vælge ud fra nogle standard action-points.

2.3 Funktions-afgrænsning

Det er nu tid til at behandle alle de funktionsforslag der er fremkommet under de mange bruger-interviews og lave en afgrænsning, sådan at det kun er funktioner af relevans, der herefter arbejdes videre med. Herunder bliver hver foreslået funktion behandlet hver for sig og der vurderes i hvert tilfælde om den teknisk og organisatorisk er mulig at implementeres. Denne afgrænsning er fremkommet i tæt samarbejde med Asger Jensen og Søren Hebsgaard (se mødereferat af 12.09.2000 bilag 8.13 side 106).

2.3.1 Kalender for samtlige medarbejdere

Dette er en meget brugbar funktion og er blevet foreslået af samtlige adspurgte brugere, men den er urealistisk af implementere rent teknisk og organisatorisk. Kalenderen ligger fysisk i en database i hovedkvarteret i USA og den er ikke mulig at få adgang til. Selv hvis det var muligt, er det helt sikkert at databasestrukturen vil blive ændret i fremtiden, da kalenderen ikke er et officielt produkt, og når den engang bliver det, vil den nok være ændret en hel del. Der er dog en anden mulighed, der kan benyttes. Man kan tage dele af kalender web-interfacet ved brug af web-stripperen i PtG, og på den måde overføre den til WAP. Men dette er et helt projekt i sig selv og vil også skulle laves om når kalenderen på et tidspunkt bliver ændret. Konklusionen er at denne funktion, selvom den er meget hot og efterspurgt, ikke er mulig at implementere og den afgrænses hermed.

2.3.2 Telefonliste over medarbejdere

En simpel funktion, dog meget brugbar og relevant, og der er adgang til databasen hvor oplysningerne ligger i. Der er dog ikke oplysninger gemt i databasen om hvilke faglige ressourcer de enkelte medarbejdere besidder. Dette kunne løses ved at oprette en lokal tabel til at gemme de enkelte medarbejderes faglige ressourcer i, men der er ikke stor sandsynlighed for at den vil blive holdt ajour, hvilket er yderst vigtigt. Grunden til at den ikke vil blive holdt ajour skyldes at chancen for at den enkelte medarbejder går ind og opdaterer i tabellen ikke er særlig stor, hvilket hænger sammen med at den ikke vil blive en del af Oracle's eksisterende system. Konklusionen er at denne funktion vil blive implementeret, dog uden mulighed for at søge efter ekspertise.

2.3.3 Time-registrering

Endnu en meget brugbar og efterspurgt funktion, men den er ligesom kalenderfunktionen ikke mulig at implementere rent teknisk og organisatorisk. Time-registreringssystemet er et lukket system. Det er ikke muligt at få adgang til databasen. Det er heller ikke muligt at tage dele af time-registrerings web-interfacet med PtG's web-stripper, da det er en java-applet. Konklusionen er at denne funktion, selvom den kunne have gjort hverdagen nemmere, ikke er mulig at implementere og den afgrænses hermed.



2.3.4 Marketings-oplysninger

En ekstrem relevant funktion, som har en meget stor værdi for sælgerne. Den database hvor oplysningerne ligger i, er der adgang til. Det er dog ikke muligt at skrive til databasen, så tilmeldinger må foregå via e-mail, som sendes til en medarbejder med fuld adgang til databasen. Det er dog ikke noget problem, for sådan foregår det også idag. Konklusionen er at denne funktion vil blive implementeret og eventuelle tilmeldinger vil komme til at foregå via e-mail.

2.3.5 Adgang til sine e-mails

Denne funktion er allerede implementeret som en selvstændig service til PtG af vores kontaktperson Asger Jensen. Det er derfor ikke nødvendigt at udvikle den igen. Asgers version kan integreres med MAOV-systemet, så brugeren ikke ser dem som to adskilte funktioner. Konklusionen er at denne funktion allerede er udviklet og derfor bare skal integreres i det endelige system.

2.3.6 Oplysninger om software-licenser

Dette er meget relevante oplysninger at have og det er muligt at få adgang til dem. De oplysninger der vil blive tilgængelige er fortsat: Hvilke produkter har en konkret kunde, hvor mange licenser og hvilken platform de kører på. Så konklusionen er at denne funktion vil blive fuldt implementeret.

2.3.7 Prisliste (produkter og konsulent-ydelser)

Det så fra starten af ikke ud til at det ville blive muligt at implementerer denne funktion, da det rent teknisk ikke var muligt at få adgang til de relevante data. Det viste sig dog senere at Christian Fabricius fra salgsafdelingen gerne ville stå for at vedligeholde en en database med prislisten, som MAOV-systemet får adgang til. Dvs. hvergang den eksisterende prisliste bliver opdateret, så sørger CF for at kopier disse data over i prislisten, som MAOV-systemet har adgang til. Prislisten kommer kun til at indeholde priser på produkter, og ikke konsulent-ydelser, som der ellers også var udtrykt ønske om. Grunden til at konsulent-ydelserne ikke er med i prislisten er at det er en kompliceret sag at beregne dem, med mange faktorer og vurderinger der spiller ind. Det er altså ikke bare en fast pris ligesom det er tilfældet med produkt-licenser. Så konklusionen er at denne funktion alligevel vil blive implementeret, dog uden priser på konsulent-ydelser.

2.3.8 Kontaktliste over kunder/partnere

En meget praktisk funktion, der er yderst brugbar når man er ude af huset. Databasen med oplysningerne er lige til at få adgang til, og den bliver løbende opdateret af Human Ressource afdelingen (HR). Så konklusionen er at denne funktion vil blive fuldt implementeret.

2.3.9 Oplysninger om aktuelle sager (TAR)

Dette er en meget værdifuld funktion at have til rådighed, da man med denne funktion kan være progressiv overfor kunden, dvs. være på forkant med eventuelle aktuelle sager. Informationerne der skal bruges for at denne funktion kan blive en realitet, er der adgang til. Så konklusionen er at denne funktion vil blive fuldt implementeret.



2.3.10 Adgang til et udsnit af "Sales Online" systemet

Der er et stort behov for denne funktion og den vil uden tvivl have stor forretningsmæssig værdi, men det er ikke muligt at få adgang til den relevante database. Der er dog en anden mulighed, der kan benyttes. Man kan, ligesom med kalenderfunktionen, tage dele af "Sales Online" web-interfacet ved brug af web-stripperen i PtG, og på den måde overføre det til WAP. Men dette er også et helt projekt i sig selv. Konklusionen er at denne funktion, selvom den ville have haft stor forretningsmæssig værdi, ikke er mulig at implementere og den afgrænses hermed.

2.3.11 Har kunden betalt for tidligere køb

Denne funktion er ikke relevant for Oracle. De kender deres kunder (navngivne kunder), og de bliver kreditgodkendt inden de kan oprettes som kunder hos Oracle. Endvidere er det jo software Oracle sælger, så der er ikke noget materielt tab hvis en kunde ikke betaler. Konklusionen er at denne funktion ikke er relevant at implementere og den afgrænses hermed.

2.3.12 Tildele action-points til kolleger

Dette bliver muligt ved at sende e-mails til hinanden. Man skal dog ikke skrive sit action-point som e-mail manuelt, men bare vælge et action-point i en menu. Resten vil så automatisk blive udfyldt. Det laves sådan at man selv modtager en e-mail (cc), så man kan huske hvilke action-points man har tildelt hvem. Konklusionen er at denne funktion vil blive fuldt implementeret.

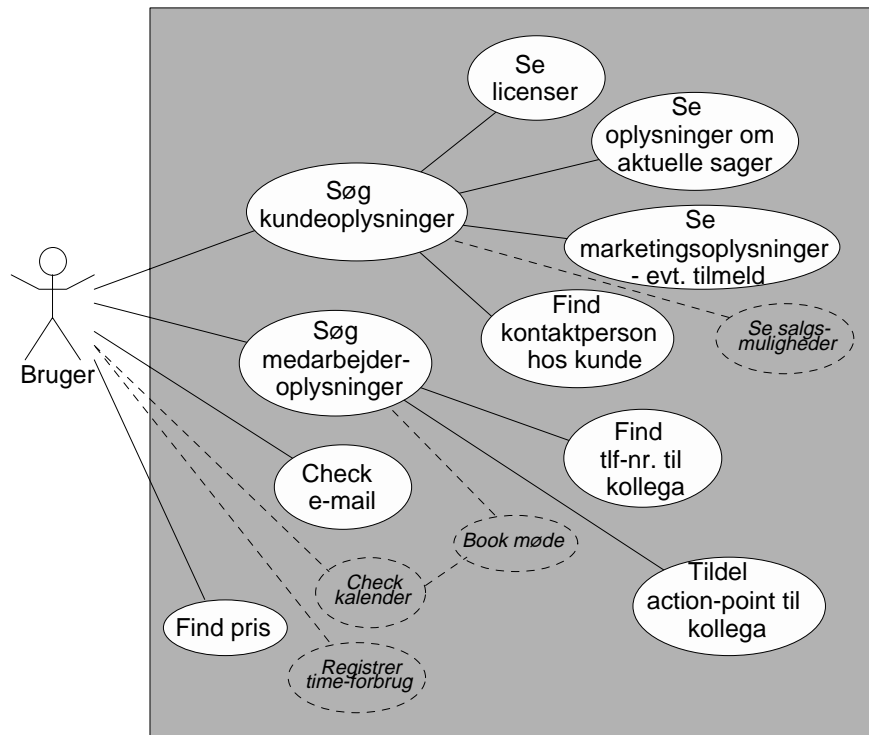
2.3.13 Konklusion på funktions-afgrænsningen

Efter denne gennemgang af hver enkelt foreslået funktion kan det konkluderes at der er 8 funktioner tilbage, som det er relevant at arbejde videre med. Det skal ikke forstås på den måde at de funktioner, på nær en, som er blevet afgrænset ikke var relevante. De er meget relevante, men kan bare ikke lade sig gøre at implementere på nuværende tidspunkt. De vil dog senere kunne tilføjes systemet, hvis de relevante data bliver tilgængelige.



2.4 UseCase-model

UseCase-modellen er udarbejdet ud fra ønskerne fra de mange bruger-interviews, og afspejler funktions-forslagene. De er struktureret for at skabe et bedre overblik. De usecases der er tegnet med stiplede, bygger på de funktionsforslag som ikke vil blive implementeret. De er taget med for at give et samlet overblik over hvordan systemet ville have set ud, uden de tekniske og organisatoriske hindringer. Bemærk at partnere opfattes som kunder.



* De stiplede dele implementeres ikke !!!

2.4.1 Aktør: Bruger

Brugeren er en generalisering af de forskellige medarbejdere fra de forskellige afdelinger. Der er nemlig ikke den store forskel på hvad de forskellige grupper skal have adgang til alligevel, da de eksterne brugere er blevet afgrænset. En bruger er udelukkende en ansat hos Oracle Danmark, primært fra salgsafdelingen. Det vil være muligt at tilpasse hvilke funktioner der skal være tilgængelige for hver enkelt bruger.

2.4.2 Søg kundeoplysninger

Mange af de ønskede funktioner omhandler (ikke overraskende) korte og konkrete oplysninger om en kunde. Det vil typisk være før et kundemøde, hvor sælgeren lige skal informeres om kunden, så han 100% ved hvad de skal holde møde om. Sælgeren snupper sin mobiltelefon og logger ind på MAOV og vælger at han vil have kundeoplysninger. Der vælges hvilken kunde det drejer sig om. De aktuelle kunder vil være ligetil at vælge, mens kunder som ikke er i brugerens kunde-portefølge, skal søges frem. Når den aktuelle kunde er fundet, kan han vælge at se de ønskede oplysninger, hvilket er beskrevet i de følgende under-usecases. Da de kun handler om at finde oplysninger, kan brugeren stoppe brugen af systemet på hvilket som helst tidspunkt ved at afbryde forbindelsen.



2.4.3 Se licenser

Det er grundlæggende simple oplysninger om licenser der har relevans. Når den aktuelle kunde er valgt, vil man kunne se en komplet liste over hvilke Oracle-produker kunden har licens til. For hvert produkt vil man kunne se, hvor mange personer der er købt licens til, og hvilken platform systemet kører på. Det kan også bruges, hvis en kunde specifikt spørger om sine licenser, hvis de har glemt det.

2.4.4 Se oplysninger om aktuelle sager

Oracle har et system, hvor de opretter en såkaldt "TAR", der løbende opdateres hver gang kunden henvender sig. Man kan deri læse hvilke problemer kunden har haft, og hvordan de er blevet løst. Man kan hurtigt slå op om kunden har et aktuelt problem, som man lige skal finde ud af om der er fundet en løsning på. På aktuelle problemer er det endvidere muligt at se hvilken prioritet problemet har, og hvem af parterne der venter på svar fra hvem.

2.4.5 Se marketingsoplysninger - evt. tilmeld

Når man har fundet sin kunde på systemet, kan man se hvilke seminarer og mailinglister kunden er tilmeldt og har deltaget i. Så ved man hvad man kan referere til under kundemødet. Man kan desuden se kommende seminarer, som kan have interesse for kunden. Hvis man finder et seminar der har interesse, kan man tilmelde kunden over WAP-telefonen. Det vil ikke være muligt at læse mere dybdegående om de forskellige seminarer, for displayet egner sig ikke til så store mængder tekst. Man ved dog hvad de går ud på ud fra navnet alene.

2.4.6 Find kontaktperson hos kunde

Man kan få en besked om at man skal kontakte en given kunde, men kan ikke huske telefonnummeret. Man slår kunden op på MAOV-systemet, og vælger kontakt-oplysninger. Her vil man kunne se hvem det er man skal have fat i hos den givne kunde, og hvordan (tlf, e-mail, adresse). Nogle kunder er så store, at der er flere kontakt-personer. Man skal derfor også kunne se hvilken rolle eller stilling de forskellige kontaktpersoner har, så man kan vælge den rigtige. Det ville være smart hvis man bare kunne vælge en person, og så ringer WAP-telefonen op direkte, men det er vist endnu ikke teknisk muligt.

2.4.7 Søg medarbejderoplysninger

Der er flere funktioner der omhandler kolleger. Man skal derfor kunne finde en vilkårlig medarbejder fra Oracle DK frem. Systemet skal huske hvilke kolleger man tidligere har fundet frem, da det ikke altid vil være nye kolleger man skal finde.

2.4.8 Find tlf-nr. til kollega

Dette er en simpel telefon-liste, der er ikke så meget ved den. De telefonnumre man ofte bruger er typisk gemt i mobiltelefonens telefonliste, så MAOV-versionen skal kun bruges, når man skal bruge et nummer til en kollega man ikke normalt har kontakt med. Man søger en person ud fra navnet, men det ville være en fordel at kunne søge folk på deres faglige kvalifikationer. Eksempelvis hvis man skal bruge en ekspert til at løse et problem, men ikke kender nogen der kan. Men dette bliver desværre ikke muligt, jævnfør funktionsafgrænsningen.



2.4.9 Tildel action-point til kollega

Denne use-case er ikke tænkt som en stand-alone. Den vil kunne knyttes til de forskellige andre use-cases. Man tildeler action-points ved at sende en e-mail hvori der står hvad der skal gøres. Eksempelvis kan man ikke tilmelde kunder til seminarer online, men man kan sende en e-mail til HR-afdelingen som så tilmelder kunden. Man skal ikke formulere denne e-mail hver gang, den skal bare vælges som et menu-punkt fra en menu. Systemet fylder selv ud hvem det er der skal tilmeldes (fordi man har valgt en kunde, når man ser efter seminarer). Det vil være nogle standard-action-points man kan vælge imellem, så man ikke skal indtaste beskeden manuelt hvergang.

2.4.10 Check e-mail

Eksempel: En konsulent skal ind til et kundemøde om et problem. En anden er sat til at løse problemet, og skal sende en e-mail om løsningen. Den var ikke nået frem inden konsulenten tog hjemmefra, men skal nu lige vide om den er kommet før kundemødet. I MAOV-systemet kan han se om den er kommet, læse den og være 100% opdateret om hvordan problemet løses. Meget af kommunikationen hos Oracle foregår via e-mails, og medarbejderne bør kunne checke deres indbox hvorsomhelst og nårsomhelst, nemt og hurtigt.

2.4.11 Find pris

Der vil altid være en aktuel prisliste over samtlige af Oracle's produkter online, og i den er det muligt ud fra nogle produktkategorier at vælge de enkelte produkter og få en aktuel pris på det.

2.4.12 De stiplede

Se salgsmuligheder: En sælger bliver ringet op og får at vide at han skal ud til en kunde. Han slår op på kunden for at se hvilke produkter han har mulighed for at sælge.

Check kalender: Der skal bruges en ekspert til et møde, hvor konsulent, ekspert og kunde deltager. Der skal findes en tid i kalenderen hvor begge kan.

Book møde: Reserver tid til mødet hos den førnævnte ekspert.

Registrer time-forbrug: Man vælger jævntligt hvilket projekt man arbejder på og registrere de aktuelle timer, frem for at skulle indtaste en hel uges arbejdstimer sidst på ugen.



2.5.5 Kunde

En kunde er en person/organisation der køber varer af Oracle Danmark.

2.5.6 TAR

En TAR bliver oprettet på en kunde når denne har et problem med f.eks en database. I TAR'en registreres løbende hvilken prioritet problemet har, hvad det drejer sig om og hvem der venter på hvem, dvs. er det Oracle der venter på yderligere oplysninger eller er det kunden der venter på svar. TAR'en opdateres hver gang der sker noget nyt i sagen.

2.5.7 Licens

En licens købes af en kunde for at få lov til at bruge et Oracle-produkt (software). Licensen beskriver bl.a hvor mange brugere der er licens til, hvilket produkt det drejer sig om og hvilken platform systemet kører på.

2.5.8 Mailingliste

En mailingliste er det en kunde kan være tilmeldt for at få bl.a nyhedsbreve om produkter eller oplysninger om diverse seminare.

2.5.9 Seminar

Et seminar er det en kunde kan tilmelde sig for at tilegne sig ny viden om f.eks et nyt produkt han påtænker sig at købe. Tilmelding foregår pr. e-mail.

2.5.10 Kontaktliste

Kontaktlisten består af kontaktpersoner hos samtlige kunder og partnere. Desuden kender kontaktlisten den enkelte medarbejders aktuelle kunder, så de er nemmere at finde.

2.5.11 Partner

En partner er et firma der har tilknytning til Oracle Danmark som partner, som f.eks sælger løsninger der bygger på Oracle-teknologi.

2.5.12 Produkt

Et produkt er et stykke software som Oracle sælger.

2.5.13 Pris-liste

En pris-liste indeholder priser på samtlige produkter.

2.5.14 De stiplede

Kalender: Hver medarbejder har sin egen kalender.

Time-seddel: Nogle medarbejdere har en time-seddel der skal indrapporteres ugentligt.

Faglige ressourcer: Faglige ressourcer er de evner der kan bruges forretningsmæssigt.

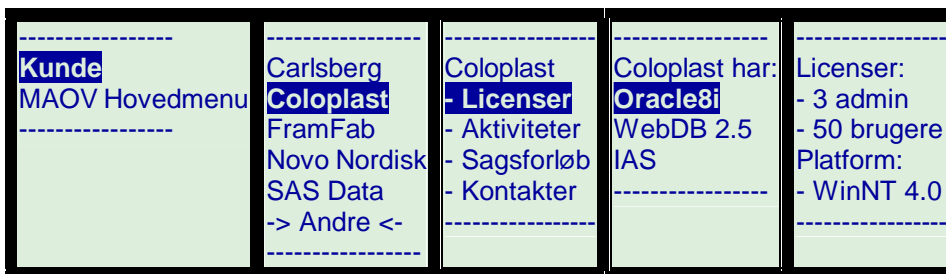
Opportunity: En opportunity er en mulighed for salg af et enkelt produkt til en given kunde.



2.6 Mock-ups

Systemet skal primært vises på en WAP-telefon, og vi har derfor designet vores mock-ups ud fra de begrænsninger det stiller. WAP-telefonens display kan kun vise et meget begrænset antal linjer på skærmen samtidigt (typisk 4), og linjernes længde er også stærkt begrænset (ca. 20 karakterer). Man kan ikke fastlægge linje-længden, da den kan variere fra telefon til telefon. Det er heller ikke alle bogstaver der er lige brede. En simpel test viste, at der kan være 40 i'er men kun 11 m'er på een linje.

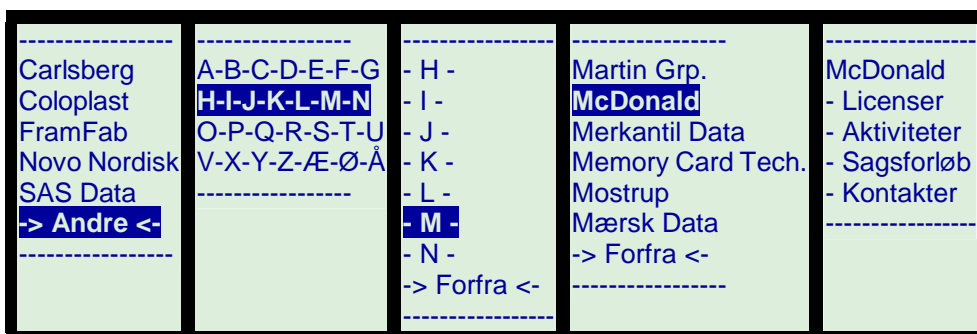
At bruge WAP er som at bladere rundt i en række menuer, med lidt tekst og menupunkter. Vi har lavet et par simple mockups der viser hvordan man kan se oplysninger om en kundes licenser. Det skal læses som en tegneserie, hvor man trykker på det punkt der står med fremhævet skrift for at komme videre til det næste!



Det første eksempel viser hvordan man finder licenser for en af de kunder man arbejder med for tiden. Systemet vil vide hvilke kunder hver enkelt bruger har gang i for tiden. Nedenunder vises hvordan man finder en kunde som systemet ikke ved man har gang i. Man søger blandt samtlige af Oracle's kunder, ved at skrive en del af navnet.



Det kan være at søge-funktionen skal ændres, så man ikke skal taste en del af navnet ind, men derimod at skulle snævre sig ind til det endelige gennem flere valg. Det kan vise sig at være mere brugervenligt på en WAP-telefon, når nu indtastningsmulighederne er så ringe.

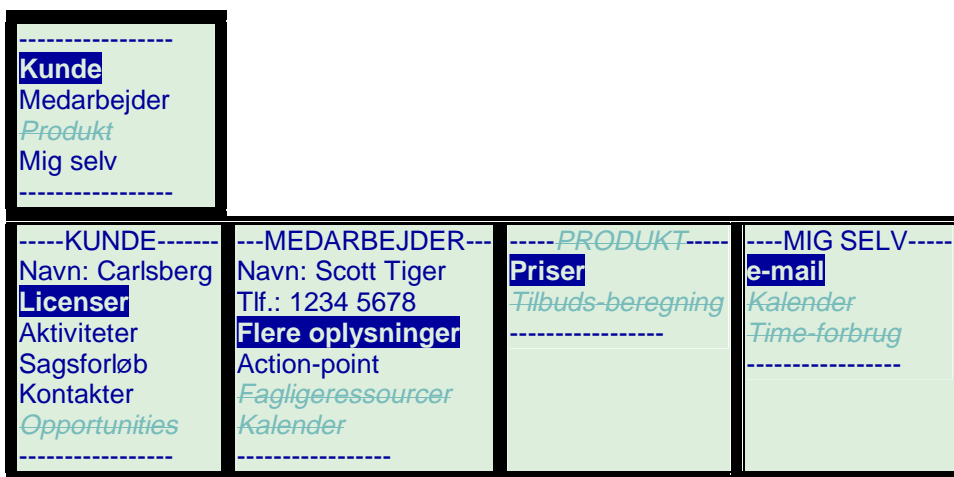


2.6.1 Menustrukturen



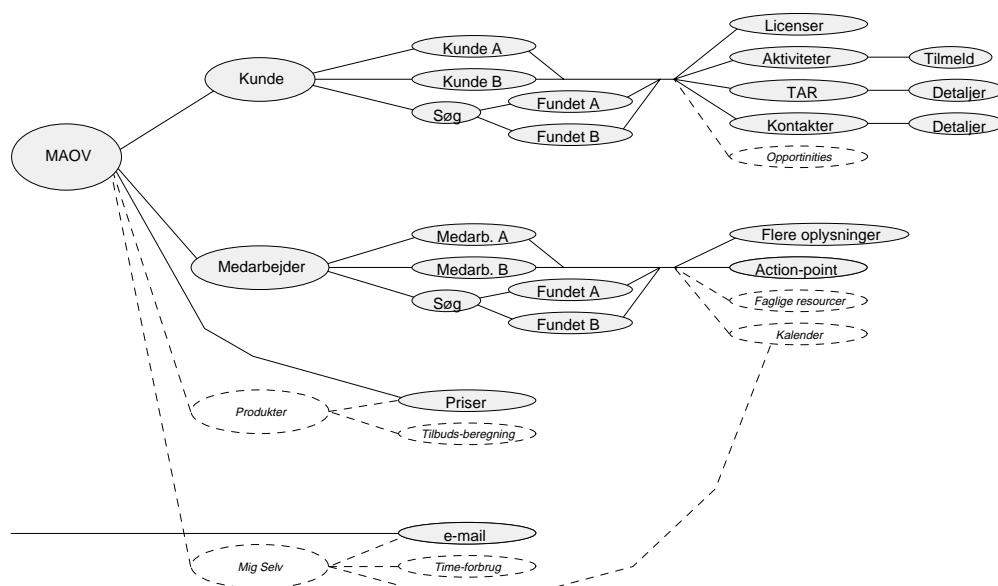
Herunder vises hvordan vi forestiller os at menustrukturen opbygges i MAOV. Når brugeren logger ind på systemet præsenteres han for en hovedmenu, hvorfra fire undermenuer kan vælges. Fra disse undermenuer er det så muligt at komme længere ned i hierarkiet. Læg mærke til at inden man kommer til kunde/medarbejder undermenuen skal der først vælges en kunde/medarbejder (vælges på en af ovenstående måder). De punkter der er fremhævet er de første man kan vælge i den enkelte menu, dvs. det der står ovenover dette punkt er ren tekst og kan ikke vælges.

Alle foreslåede funktioner er taget med for at give et samlet overblik over hvordan systemet ville komme til at se ud, hvis ikke der var de tekniske og organisatoriske hindringer. De ikke mulige funktioner er streget ud.



2.6.2 Navigations-diagram

For nemmere at overskue strukturen i MAOV-systemet har vi lavet en navigations-diagram, der viser hvordan man navigerer rundt i det. Igen er alle foreslåede funktioner taget med for at give et samlet overblik over hvordan systemet ville komme til at se ud. De ikke mulige funktioner er illustreret med stiplede linjer. Denne menustruktur kan sammenlignes med MAOV-systemets forgænger (Oracle's prototype), som er beskrevet i bilag 8.3 side 87.





2.7 Konklusion på kravmodel

Vi havde henvendt os til 15 medarbejdere her hos Oracle, fik svar fra 10 og interviewede 6. Heri ikke indberegnet vores to kontakt-personer, der har et godt indblik i hvad der er krævet af MAOV-systemet. Vi har været rundt i alle afdelinger på nær support-afdelingen, hvor ingen reagerede på vores henvendelser, og det så umiddelbart ud til at være et problem. Asger, vores ene kontakt-person kommer dog fra den afdeling, så han kender arbejdsgangen.

Ud fra de udførte bruger-interviews har vi fået en række ønsker om funktioner, hvoraf knap 1/3 viste sig at være umulige at opfylde. Oracle tillod ganske enkelt ikke adgang til de relevante data. Der er dog stadig rigeligt med funktioner tilbage til at kunne retfærdiggøre den videre udvikling af MAOV-systemet. De tilbageværende funktioner er desuden nogen der vil give stor forretningsmæssig værdi for Oracle Danmark.

Brugergrupperne er reduceret til een universel brugergruppe, idet de grupper som adskilte sig væsentligt fra resten (kunder og partnere), er afgrænset, jævnfør sikkerhedsmæssige forhold.

Under interviewet med Christian Bak d. 30.08.2000 (bilag 8.14 side 114) fra økonomi-afdelingen, viste det sig at projekt MAOV strider imod virksomhedens globale IT-strategi, der forbyder egen-udvikling udenfor hovedkvarteret i Californien. Læs mere om Oracle's globale IT-strategi i bilag 8.2 side 85.

Funktionerne er fastlagt, og der kan ikke komme flere til. Det er specificeret hvad brugerne vil have, samt de tekniske retningslinjer er udarbejdet. I det efterfølgende afsnit bliver systemet modelleret i en analysemodel.



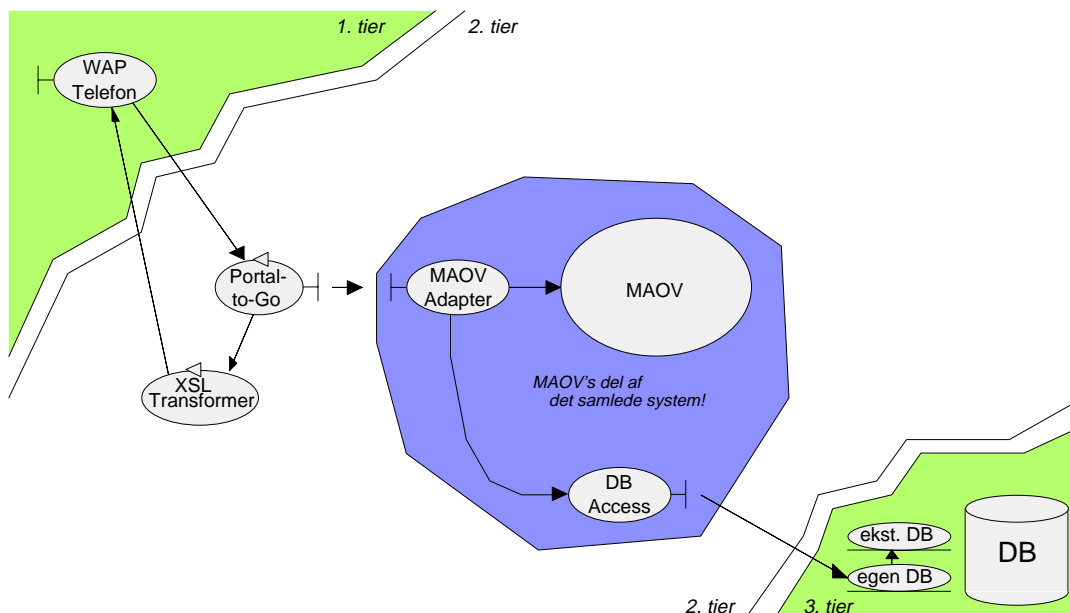
2.8 Indledning til analysemodel

En analyse-model er normalt modelleret ud fra den holdning at alt kan lade sig gøre (den ideelle verden), uden nogen form for teknologiske begrænsninger. MAOV-systemet er dog allerede fra start af fastlåst af en række teknologi-valg, og derfor er analysemodellen blevet modelleret så den er tilpasset disse. Læseren forventes at have en grundlæggende indsigt i den objektorienterede verden, for at forstå dette afsnits modeller.

Analysemodellen indeholder normalt to modeller: En ICE-model og en udvidet/intern usecase-model. Den interne use-case er dog udeladt, da den kun består af een use-case. MAOV-systemets direkte bruger er PtG, og PtG er fuldstændig ligeglad med hvad MAOV-systemet gør. PtG skal bare have nogle XML-sider fra MAOV-systemet, som den kan formattere og sende videre til slut-brugeren. Det er også derfor at MAOV-systemet kun har eet interface-objekt til PtG på ICE-modellen.

2.9 ICE-model

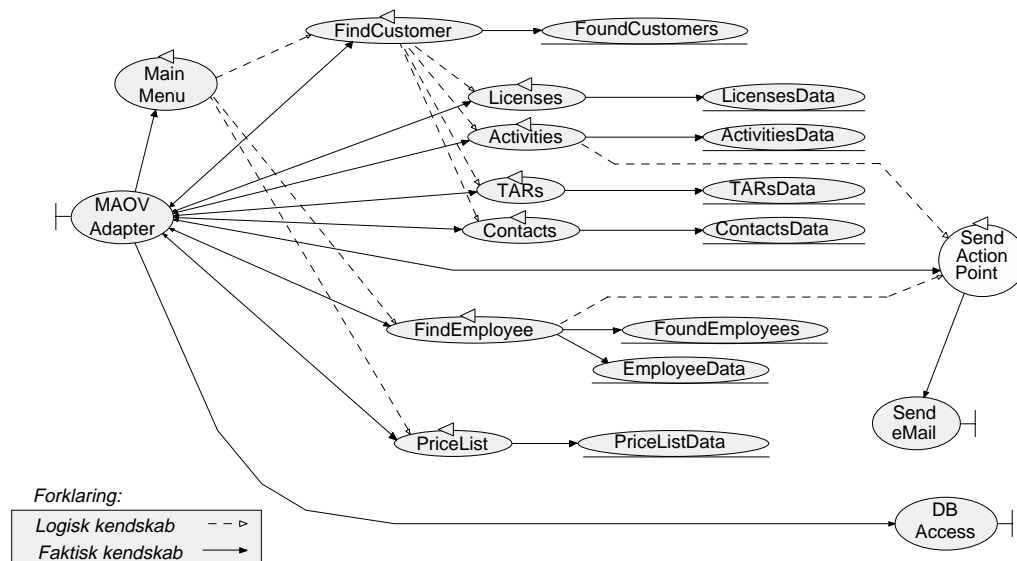
ICE-modellen beskriver de forskellige dele af systemet og hvem der kommunikerer sammen. For at give et overblik over systemet vises her først det samlede system fra WAP-telefon gennem PtG og et forsimplet MAOV-system, til de data der ligger i databasen. Desuden beskrives hvilke tier's de forskellige dele ligger på. MAOV ligger selv kun på tier 2, dog med nogle egne tabeller på tier 3.



Brugeren står med WAP-telefonen og kalder op til Portal-to-Go. Forespørgslen sendes videre til MAOV-systemet, som henter de ønskede data fra databasen. Data'ene sendes tilbage til PtG som XML, der sendes igennem XSL-Transformeren til WAP-telefonen. Det er værd at bemærke at alle kald til MAOV-systemet går til MAOVAdapteren. Det er altså dens opgave at sende opgaven videre så den kan blive løst.



Internt i MAOV er systemet delt op efter funktioner. På den måde er det nemt at tilføje nye funktioner, da de andre kan forblive uændrede. De to objekter: MAOVAdapter og DBAccess er de samme på begge tegninger. Det er det store MAOV-objekt, der er delt ud på nedenstående figur.



Eftersom systemet kører som en Adapter, er de faktiske kendskabs-relationer ikke som man skulle forvente. Det er MAOVAdapteren der skal kende alle kontrol-objekterne, som derimod ikke skal kende hinanden i koden. Der er derfor indtegnet de logiske kendskabs-relationer (stiplede linjer), så man kan se hvilke objekter der fører brugeren videre til andre objekter.

Når MAOVAdapteren modtager en forespørgsel, sendes den videre til det relevante kontrol-objekt, der så svarer på forespørgslen. Der er ikke kun en forespørgsel til hvert kontrol-objekt, så forespørgslen er delt op i en action og en sub-action (action til MAOVAdapteren, sub-action til kontrol-objektet). Se de forskellige action-værdier og forklaring dertil, i afsnit 4.5 side 57.

Når kontrol-objektet skal bruge data, opretter det et entitets-objekt der selv henter oplysningerne i databasen. Entitets-objektet får DBAccess med fra kontrol-objektet som henter det fra MAOVAdapteren. Alle entitets-objekterne har derfor reelt fat i DBAccess, men kun når de henter data fra databasen. Derfor er der ingen kendskabs-relationer mellem entitets-objekterne og DBAccess-objektet.

Herunder beskrives alle objekterne fra begge illustrationer kort.

- **WAP Telefon:** Den håndholdte enhed der viser outputtet fra PtG.
- **Portal-to-Go:** Systemet der kan formattere XML til forskellige enheder.
- **XSL Transformer:** Heri står hvordan PtG skal formattere XML'en til den respektive enhed.
- **MAOVAdapter:** Vores interface til PtG. Alle kald fra brugere/PtG går til dette objekt.
- **MainMenu:** Genererer hovedmenuen i XML.
- **FindCustomer:** Genererer de sider der guider brugeren frem til den ønskede kunde (se mock-ups afsnit 2.6 side 18).
- **FoundCustomers:** Indeholder oplysninger om hvilke kunder brugeren senest har brugt i systemet. Indeholder desuden oplysninger om de kunder der er fundet ved fritekst søgning.
- **Licenses:** Genererer de sider der viser hvilke licenser kunden har.
- **LicensesData:** Indeholder oplysninger om hvilke licenser kunden har.



- **TAR:** Genererer de sider der viser hvordan kundens aktuelle sager forløber (TAR).
- **TARData:** Oplysninger om en kundes aktuelle sager (en TAR).
- **Contacts:** Genererer sider med kontaktoplysninger for kundens/partnerens kontaktpersoner.
- **ContactsData:** Oplysninger om hvordan man får fat i kontaktpersonerne hos kunden.
- **Activities:** Genererer de sider der viser hvilke seminarer og mailing-lister kunden er tilmeldt og har været på. Desuden vil den kunne vise hvilke fremtidige seminarer der tilbydes, og give mulighed for tilmelding.
- **ActivitiesData:** Oplysninger om aktiviteter der har relation til kunden, såsom seminarer og mailing-lister.
- **FindEmployee:** Genererer de sider der guider brugeren frem til den ønskede kollega.
- **FoundEmployees:** Indeholder oplysninger om hvilke medarbejdere brugeren senest har brugt i systemet. Indeholder desuden oplysninger om de medarbejdere der er fundet ved fritekst søgning.
- **EmployeeData:** Indeholder oplysninger om een kollega.
- **SendActionPoint:** Opretter et actionpoint, og fylder op med de oplysninger der er givet i situationen. Resten kan nemt vælges. Actionpointet skal sendes som e-mail.
- **PriceList** Genererer sider med priserne for Oracle's forskellige produkter.
- **PriceListData:** Oplysninger om hvad de forskellige licenser for produkter koster.
- **Send e-mail:** Sender en e-mail.
- **DBAccess:** Opretter en forbindelse til databasen, så man kan hente informationer derfra. Bruges af entitetsobjekterne, selvom det ikke direkte fremgår af illustrationen.

- **ekst. DB:** Repræsenterer samtlige af Oracle's eksisterende databaser, som MAOV-systemet skal tilgå.
- **egen DB:** Repræsenterer de tabeller vi selv opretter til at gemme oplysninger i, der kun skal bruges af MAOV-systemet. Eksempelvis hvilke kunder der skal være direkte adgang til for den enkelte bruger.

- **1. tier:** Den håndholdte enhed. Klienten.
- **2. tier:** Server hos Oracle DK hvor PtG kører, og dermed også MAOV-systemet.
- **3. tier:** Databaserne ligger på servere i USA. Den vi selv opretter tabeller på ligger dog lokalt her hos Oracle DK.

2.10 Konklusion på analysemodel

Analysen af hvordan systemet skal opbygges er fuldendt. Det er tilpasset de systemer det skal arbejde sammen med, og de objekter systemet skal bestå af er specificeret. Den valgte teknologi har sat sit præg på modelleringen af systemet, og det har betydet at systemets faktiske kendskabs-relationer ikke følger systemets menu-struktur, som ellers er de logiske kendskabs-relationer, som man umiddelbart ville modellere systemet efter.

Om MAOV-systemets bruger er en person eller Portal-to-Go kan diskuteres. Den direkte bruger er Portal-to-Go, men PtG er kun et mellemlag der skaber forbindelse mellem MAOV-systemet og personen. MAOV-systemet kommunikerer derfor direkte med Portal-to-Go, men det der kommunikeres sendes videre til personen og skal derfor tilpasses derefter.



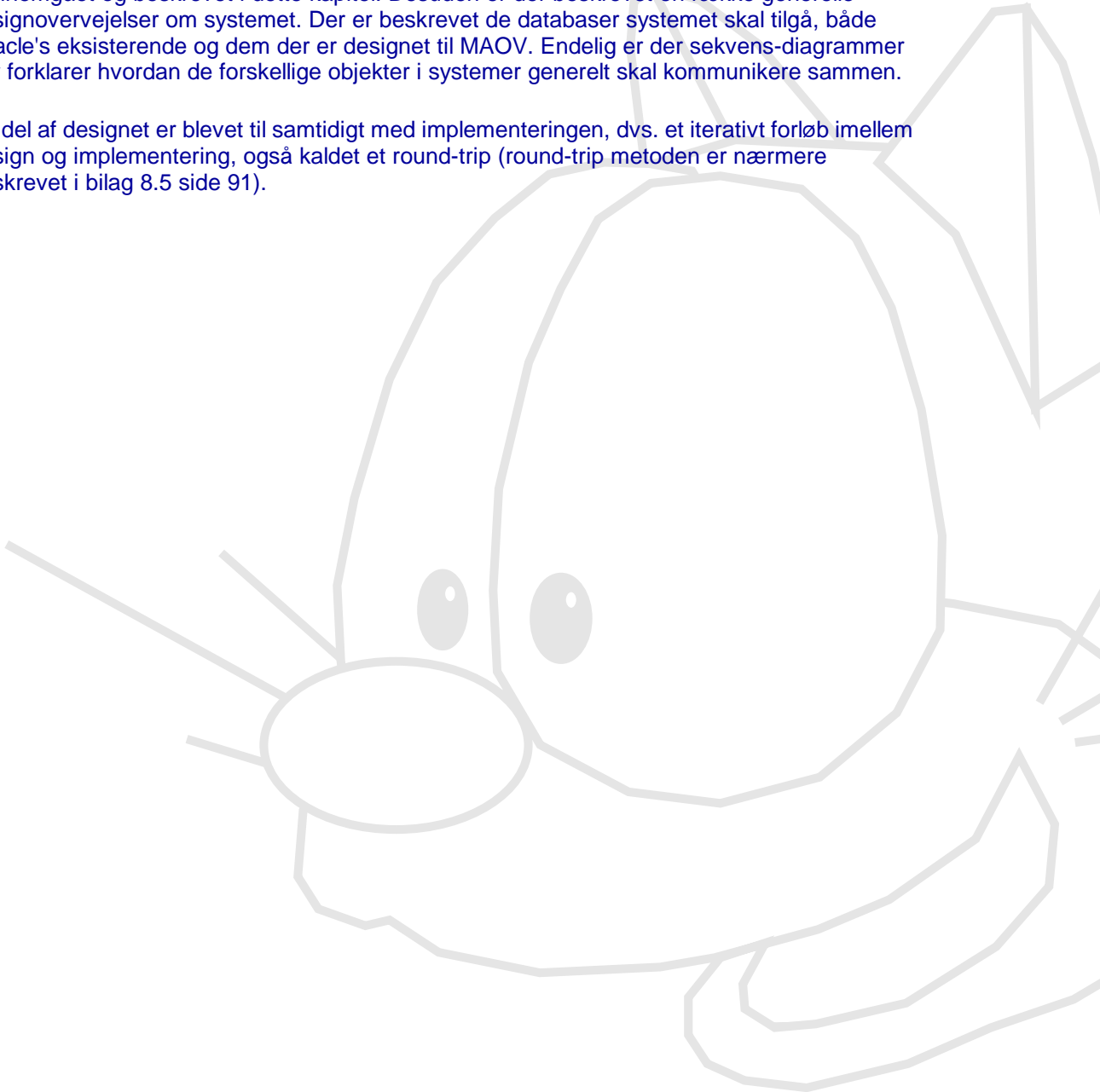
3. Design

3.0.1 Indledning

I dette kapitel vil diverse designovervejelser for systemet blive beskrevet. Kapitlet er rettet mod den mere tekniske målgruppe, og kan læses af folk der vil vide hvordan systemet er designet.

De forskellige valg af teknologi er foretaget allerede da projektet blev igangsat. De er dog gennemgået og beskrevet i dette kapitel. Desuden er der beskrevet en række generelle designovervejelser om systemet. Der er beskrevet de databaser systemet skal tilgå, både Oracle's eksisterende og dem der er designet til MAOV. Endelig er der sekvens-diagrammer der forklarer hvordan de forskellige objekter i systemer generelt skal kommunikere sammen.

En del af designet er blevet til samtidigt med implementeringen, dvs. et iterativt forløb imellem design og implementering, også kaldet et round-trip (round-trip metoden er nærmere beskrevet i bilag 8.5 side 91).





3.1 Teknologi-valg

Til implementation af systemet er valgt følgende teknologier:

- Portal-to-Go
- Oracle Database
- Java
- XML
- WAP

Valg af teknologi var stort set fastlagt ved projektets start. Projektets formål er at give mobil adgang til oplysninger, og Oracle har et produkt der er beregnet til netop det. Portal-to-Go. Det er et meget åbent system der kan kombineres med en masse andre teknologier, men det sætter alligevel nogle retningslinjer.

3.1.1 Portal-to-Go

Portal-to-Go formatterer sider så de kan vises på andre enheder end en almindelig WEB-browser, deri inkluderet mobile enheder som WAP. Alternativet ville være at lave systemet i WML direkte til WAP-telefonen, men eftersom Portal-to-Go er Oracles løsning på problemet, faldt det naturligt at bruge det. Så har vi også kunnet få hjælp og vejledning til brugen af det. Portal-to-Go's mulighed for automatisk at kunne formattere siderne til et hvilket som helst format, inklusiv fremtidige, gør det også til et godt og meget fremtidssikret valg. Portal-to-Go er mere detaljeret beskrevet i bilag 8.8 side 95.

3.1.2 Oracle Database

Oracle's hoved-produkt er deres database, så også her er valget faldet naturligt. Vi skal bruge oplysninger fra bl.a firmaets egen kunde-database, som også er en Oracle-database. Desuden har vi gennem hele datamatiker-uddannelsen arbejdet med Oracle-databaser, så det passer os glimrende. Tilgangen er ganske enkelt med SQL via JDBC. Vi har ikke fundet det nødvendigt at bruge PL/SQL, da vi primært skal hente oplysninger frem med "SELECT statements".

3.1.3 Java

Java er det sprog projektdeltagerne er blevet oplært i at programmere i, så det er vores foretrukne sprog. Portal-to-Go er selv implementeret i Java, og de plugins (adaptere) man kan lave til PtG skal også være Java. Der er så lavet nogle standard-adaptere (i java) der gør det muligt at lave adaptere i andet end java, så det er egentlig ikke noget krav. Oracle DK har selv lavet et simpelt system udelukkende i PL/SQL til Portal-to-Go. Men igen, java er vores foretrukne sprog, og det passer godt til Portal-to-Go. Desuden er Java et meget udbredt og anerkendt sprog.

3.1.4 XML

XML bliver brugt af Portal-to-Go, så her er ikke rigtig noget valg. Vi kommer i berøring med det, når vi skal bygge siderne op, som PtG skal modtage. Det gøres i java ved at linke nogle objekter sammen i en træ-struktur, der danner et dokument. Vi skal altså ikke skrive XML-dokumenter som man kender dem i form af tekst med tag's, men opbygge dem i Java. Portal-to-Go har nogle hjælpe-klasser, så det er ikke særlig svært. Portal-to-Go bruger XSL til at formattere siderne til de forskellige enheder, men der er skrevet det XSL der er nødvendigt til WAP, så vi har ikke selv noget med XSL at gøre. Portal-to-Go's standard for XML-dokumenter er beskrevet i bilag 8.9 side 97.



3.1.5 WAP

WAP-telefonen er den eneste reelle mobile online enhed på markedet. Vi har dog ikke udelukkende valgt den, da Portal-to-Go kan formattere siderne til hvad-som-helst, og deriblandt WAP. Vi har dog udviklet systemet med WAP som den teknologi systemet skal vises på, for at få et færdigt brugbart produkt. Når det i fremtiden skal vises på andre enheder, kan Portal-to-Go sørge for det. Det er meget begrænset hvad WAP kan vise, så hvis systemet kan vises på WAP, så vil det også kunne vises på andre enheder. Derimod hvis vi udviklede til en mere avanceret standard (f.eks HTML eller Flash), ville man måske ikke kunne se alt indholdet på den mere begrænsede WAP-telefon. Hvordan WAP-telefonen får adgang til MAOV-systemet, er beskrevet i bilag 8.7 side 94.



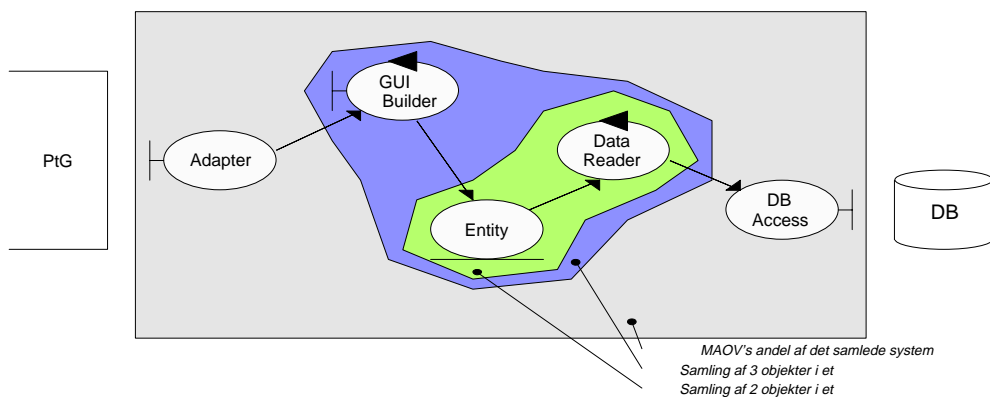
3.2 Designovervejelser

3.2.1 Indledning

Før systemet implementeres er der en række design-overvejelser der skal tages stilling til. Teknologien er fastlagt, men der er stadig mange måder grundlæggende at designe sit system på. Et godt grundlæggende design giver et mere velstruktureret og ensartet system. Det gør det nemmere at udvikle og frem for alt nemmere at ændre/vedligeholde senere hen. I det følgende er beskrevet de forskellige overvejelser vi har gjort, og en gennemlæsning bør give en bedre forståelse af kodens opbygning.

3.2.2 Størrelsen af Objekterne

Den del vi skal implementere ligger mellem Portal-to-Go og diverse databaser. Selv om der er mange funktioner i MAOV-systemet, vil hver enkelt funktion grundlæggende blive opbygget ens. Det handler om at hente data fra databasen, formatere det til XML og sende det til Portal-to-Go.



På denne model er systemet delt op i fem objekter, der hver har en meget begrænset opgave. Ved at dele systemet op i mange objekter, vil det være nemmere at tilpasse systemet, hvis det skal flyttes, da nogle af objekterne kan forblive uændrede. Eksempelvis hvis databasen ændres, så data kommer til at ligge på en anden måde, er det kun "Data Reader"-objektet der skal ændres. "DB Access"-objektet er meget generelt, og vil kun skulle ændres hvis det er en væsentlig anderledes database. Objekterne på ovenstående figur har følgende funktioner:

- **Adapter:** Interfacet til Portal-to-Go. Det er dette objekt der bliver kaldt af PtG hver gang der skal vises en side, og den indeholder en masse specifikationer om sig selv. Der er kun een Adapter som sender opgaverne videre til de repektive GUI-buildere.
- **GUI Builder:** Opbygger den XML der skal leveres til PtG. Det er for så vidt brugerfladen den bygger op, men altså i form af XML. De data den skal bruge hentes direkte fra entitetsobjektet efter behov.
- **Entity:** Indeholder de data der skal vises. Ideen er at når man instantierer entitetsobjektet henter det selv de respektive oplysningerne fra databasen, men det har vi her isoleret ud til Data Readeren-objektet, så entitetsobjektet er simpelt og rent.
- **Data Reader:** Henter data fra eksempelvis databasen, og gemmer det i det respektive entitetsobjekt. Om det skal kaldes i constructoren til entitets-objektet eller i GUI-bilderen er ikke fastlagt.
- **DB Access:** En simpel adgang til databasen. Sørger for at der er en connection til databasen, og hjælper med at sende SQL-kommandoer afsted.

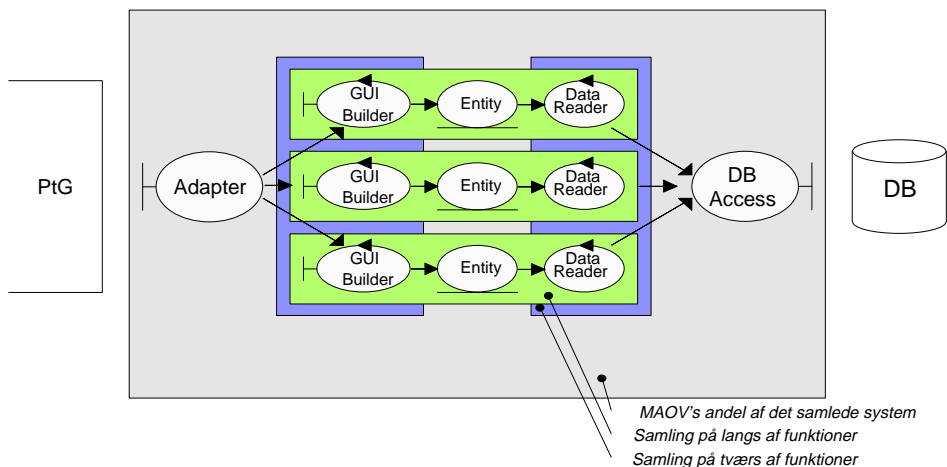


Ideelt objektorienteret, skal systemet deles op i de mange objekter, men vi kan allerede nu forudse, at flere af objekterne vil blive meget små. Det kan derfor være en fordel, at flere af objekterne smelter sammen. Nogen ændringer vil alligevel kræve ændringer i flere objekter alligevel. Eksempelvis, hvis man skal have en ny oplysning med på en side, skal Data Readeren hente den nye information, Entity-objektet skal kunne indeholde den, og GUI Builderen skal vise den. Der er to grundlæggende muligheder for sammensmeltning:

- **Samling af 3 objekter i et:** På denne måde vil en funktion blive samlet i eet objekt, og det samlede projekt vil derfor blive mere overskueligt. De tre objekter der smelter sammen vil alligevel ikke skulle bruges af andre objekter. Det vil dog blive store objekter.
- **Samling af 2 objekter i et:** Ved at skille GUI Builderen ud, vil koden blive mere overskuelig. Man vil desude opnå et smart entitetsobjekt, der selv henter oplysningerne ud af databasen, når det oprettes. Det ville begrænse bekymringerne om databasen meget, for man vil ikke udefra kunne se, hvor entitetsobjektet får sine data fra.

Vi vælger løsningen med 2 objekter i et, da det giver en logisk opdeling af systemet, så den ene halvdel koncentrerer sig om at generere XML'en der skal vises, mens den anden sørger for at hente de relevante oplysninger i databasen (layers, se bilag 8.6 side 92).

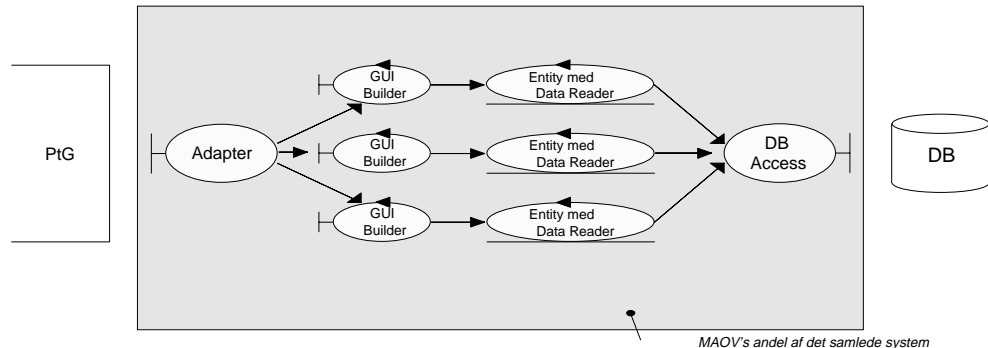
Der er en helt anden model vi har overvejet men gået bort fra. Det går ud på at der er een klasse der kan generere samtlige entitets-objekter, på tværs af funktionerne. Det vil være nogenlunde ens hvordan de forskellige entitetsobjekter hentes fra databasen, så det kunne være en fordel at samle dem. Dette er illustreret på nedenstående model.



- **Samling på langs af funktioner:** Sådan vi vil implementere systemet. Hver funktion er isoleret, så det ikke vil påvirke fungerende funktioner hvis nye tilføjes, hvilket er sandsynligt i systemet.
- **Samling på tværs af funktioner:** På den måde samles de funktioner der minder om hinanden, og derfor skal programmeres nogenlunde ens. Hver funktion vil så få en enkelt metode i hver af de delte objekter, frem for at have en masse objekter med kun een metode i hver. Spørgsmålet er så om der kun vil komme en metode i hver af de små objekter. At samle objekterne på denne måde vil gøre det besværligt at tilføje nye funktioner, da stort set samtlige objekter skal ændres, med risiko for at de ikke kommer til at virke mere.



Konklusionen af disse overvejelser er, at vi samler Entity og Data Reader i eet objekt for hver funktion. Der vil blive een Adapter som fordeler opgaverne ud, og kun een DB Access der bruges af alle entitets-objekterne. Hvis der skal tilføjes nye funktioner så er det kun Adapteren der skal kende den nye funktion, ingen af de andre objekter skal modificeres. På nedenstående figur er afbilledet tre vilkårlige funktioner.



Entitets-objekter henter selv deres data fra databasen når de oprettes. Dataene gemmes i instansvariabler i entitets-objektet, så de er lige til at få fat på, med get-metoder.

3.2.3 Public variabler vs. get- og setmetoder

Det er anset som god skik at lave get- og setmetoder, frem for bare at definere sine variabler som public, men det er nemmere og mere direkte at arbejde med public variabler. Get- og setmetoder anbefales for at beskytte sine variabler. Man udelader sin setmetode for at skrivebeskytte variablen, men det er ikke ligegyldigt om man returnerer en primitiv variabel, eller en reference til et objekt.

Returnerer man en primitiv variabel (int, char, float o.s.v) er den oprindelige variable beskyttet, da det kun er en kopi af variabelens indhold der returneres. Returnerer man derimod et objekt eksempelvis en Vector, returnere man en reference til selve objektet, og modtageren kan så frit ændre i objektet med de metoder der er i klassen som addElement(Object) og clear() i eksemplet med en Vector. Man bør derfor ikke returnere objekter med get-metoder.

String er også et objekt, og man bør derfor umiddelbart ikke returnere String-objekter. Men String-objektet er specielt idet at det ikke kan ændres. En metode som eksempelvis substring som skærer noget af en String ændrer ikke i selve den String man udfører den på, men derimod oprettes en ny String med resultatet af substring-kaldet og returnerer den. Derfor er den oprindelige String beskyttet, og man kan derfor uden bekymring returnere String-objekter i get-metoder.

Et array oprettes ikke helt som et objekt, men det er lige så ubeskyttet som eksempelvis en Vector. Hvis man returnerer referencen til et array kan modtageren frit indsætte falske værdier på de forskellige pladser i array'et. Man bør derfor give array-indexet med til get-metoden, som så kun returnerer den enkelte primitiv eller String som er på den plads i Array'et. Der er dog ikke noget galt i at returnere et helt array som er oprettet og fyldt med primitiver/Strings i get-metoden, hvis det kun skal returneres, og ikke gemmes i objektet der lavede det. Hvis der bliver sat falske værdier i et sådant array, vil det ikke påvirke objektet der oprettede det.

MAOV-systemet stiller ikke de store krav til at beskytte data, som ovenfor beskrevet. Men når man bruger tid på at lave get-metoder kan man lige så godt gøre det ordentligt. Erfaringerne for dette afsnit bygger på forsøgene i klassen TestPassingObjects (også beskrevet i afsnit 5.2 om design-test side 64).



3.2.4 Genbrug objekter vs. opret nye

Systemet består af flere objekter af forskellige klasser. Man kunne oprette samtlige objekter, når programmet startes, men det er tidskrævende, og det er ikke nødvendigvis alle objekterne der skal bruges. Derfor er det smartere, kun at oprette objekterne når de skal bruges. Men hvad gør man så når man skal bruge dem flere gange.

Enten opretter man dem hver gang man skal bruge dem. Det tager tid, og er unødvendigt, hvis indholdet ikke skal ændres i objektet. Man kunne istedet undersøge om de er oprettet hver gang de skal bruges, og så kun oprette dem, hvis de ikke allerede er oprettet. Det er simpelt at programmere:

- `if(entity == null) entity = new Entity(arguments);`

Entitets-objekterne henter selv deres indhold ud fra databasen når de oprettes, eksempelvis oplysninger om licenser for et firma. Hvis brugeren vil se licenser for et andet firma, skal entitets-objektet opdateres med oplysninger om det andet firmas licenser. Spørgsmålet er så om objektet skal have en update-funktion, som gør det samme som constructoren, eller om constructoren skal kalde update-funktionen?

Vi har valgt ikke at kunne opdatere et entitets-objekt. Man opretter objektet når man skal bruge det, og opretter et nyt, hvis man skal bruge et med nyt indhold, men af samme type. Det giver ingen mening at oprette et af vores entitets-objekter uden indhold, så constructeren skal indeholde den kode der henter data op fra databasen.

Noget af det der tager relativt lang tid i java er oprettelse af objekter og garbage-collection, men om vi vælger at genbruge frem for at oprette nye af vores objekter kommer ikke til at betyde noget mærkbart i vores samlede system, da der bliver oprettet mange objekter af de andre java-klasser vi bruger (PtG, java.sql). Vi kunne godt lide ideen med at entitets-objektet selv fylder sig med data, og det virker mest logisk at man opretter et nyt, når man skal have nye data.

Det har vist sig under implementationen at det ikke har været alle informationer man kan hente fra databasen ved oprettelse, så de hentes senere med et metodekald. Hvis vi skulle følge ideen med kun at hente oplysninger i constructoren, skulle vi have lavet endnu flere entitets-klasser. Vi mente dog at yderligere klasser ville gøre systemet unødvendigt uoverskueligt.



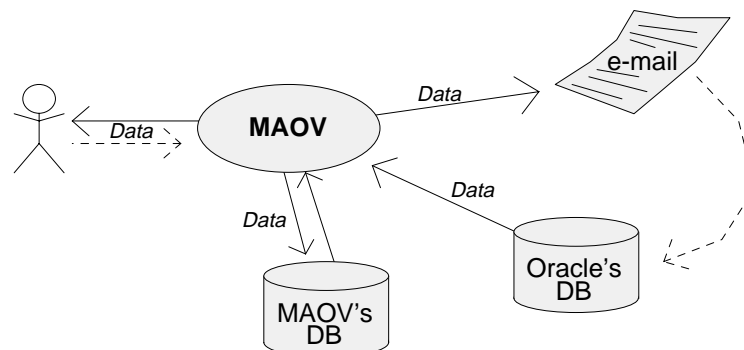
3.2.5 Hvad læses, og hvad skrives

Systemets primære opgave er at hente (læse) data fra Oracle DK's databaser, og præsentere dem for brugeren på en håndholdt enhed (WAP). MAOV-systemet kan på ingen måde få skrive-rettigheder til Oracle's centrale databaser, da projektet som skole-projekt ikke kan opnå nok tillid. Det er dog ikke noget problem, da vi kun skal læse de centrale oplysninger.

Det har ikke været muligt at opnå læserettigheder til samtlige af Oracle DK's centrale databaser, og det har i sig selv begrænset systemet. Det har faktisk været grunden til de fleste afgrænsninger af systemet. Der er dog rigeligt tilgængelig data til at lave et brugbart MAOV-system.

Det skulle være muligt at tilmelde kunder til seminarer med MAOV-systemet, hvilket umiddelbart kræver at vi kan skrive til databasen. Det er dog ikke tilfældet, idet tilmelding til seminarer kun kan udføres af HR-afdelingen, af organisatoriske årsager. MAOV-systemet skal derfor sende en e-mail til HR, som foretager skrivningen i databasen. Med andre ord, skrivning til databasen foregår ved at sende en e-mail til en der har skrive-rettigheder. Det virker besværligt at skulle blande et menneske ind i systemet, men det er den måde det skal foregå hos Oracle DK.

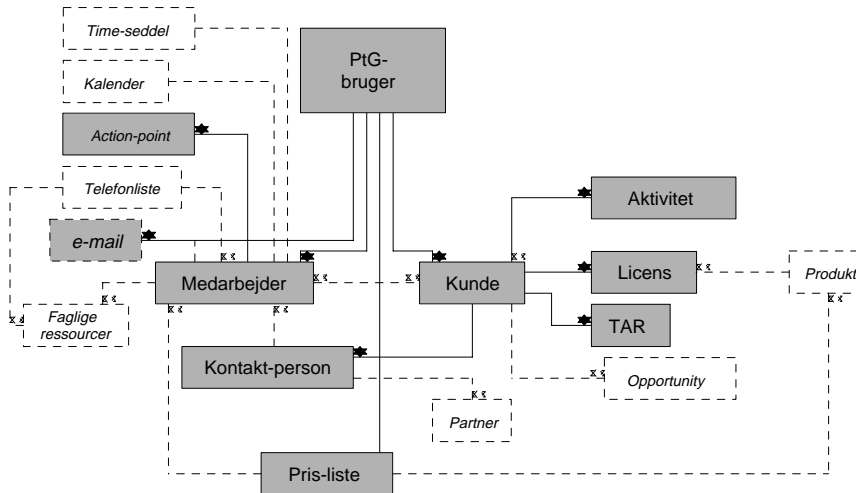
MAOV-systemet har adgang til en database med fuld læse/skrive-rettigheder. I den kan MAOV-systemet gemme data, der kun skal bruges til MAOV-systemet. Det kan være oplysninger om hvilke kunder en bruger sidst har søgt på, så brugeren ikke skal søge de samme kunder igen og igen. Desuden vil de mulige action-points blive gemt i en tabel, så nye kan tilføjes uden at koden skal re-compile.





3.3 Tilrettet Domæne Model

Domæne Modellen, som beskrevet i analysen, viste sig ikke at hænge sammen med de databaser MAOV-systemet kan tilgå. Det var forventet at Oracle DK's databaser indeholdt alle tænkelige attributter og links, hvilket ikke var tilfældet. Domæne Modellen er derfor blevet tilrettet, så den bedre beskriver domænerne i det faktiske system som det blev implementeret.



* De stiplede dele kunne ikke lade sig gøre !!!

Den nye model er sammelignelig med den gamle, idet de objekter der er forblevet uændret, har fastholdt deres pladser på figuren. De objekter og links der ikke var mulige at have med, er tegnet med stiplede linjer, ligesom dem der var afgrænset allerede på den oprindelige domæne-model. Et enkelt nyt objekt er blevet tilføjet. (oprindelig domæne model, se afsnit 2.5 side 16)

3.3.1 PtG-bruger (ny)

Portal-to-Go holder styr på sine forskellige brugere. PtG-brugeren er derfor et domæne der er relevant for MAOV-systemet. Hver enkelt PtG-bruger har sit eget login. MAOV-systemet husker hvilke kunder/medarbejdere de enkelte brugere har søgt frem, og det skal derfor vide hvilken bruger der er på for at vise netop den brugers sidst søgte kunder/medarbejdere, og ikke de andres. PtG-brugeren bruger MAOV-systemet som en ud af flere services. Der kan derfor også være forskellige opsætninger af services til hver enkelt bruger.

3.3.2 Medarbejder

En medarbejder er stadig en person der arbejder hos Oracle Danmark. Imod forventning er der dog ikke noget link til hvilke kunder den enkelte medarbejder har.

3.3.3 e-mail (ændret)

Oracle's medarbejdere har hver en e-mail konto. Det er de nye meddelelser i inboxen, der har relevans for MAOV-systemet. Rammen udenom e-mail er stiplede fordi det ikke er en direkte del af MAOV-systemet, men en selvstændig service som PtG-brugeren kan bruge.



3.3.4 Action-point (ændret)

Man kan ikke se hvilke action-point der er tilknyttet en medarbejder. Til gengæld kan man tildele action-points til andre medarbejdere, som de så får i deres email-box.

3.3.5 Kunde

En kunde er stadig en person/organisation der køber varer af Oracle Danmark.

3.3.6 TAR

TAR er helt uændret.

3.3.7 Licens

Grundlæggende er licens-objektet ikke ændret, men det har mistet sit link til produktet, som kunne linke videre til en pris.

3.3.8 Aktivitet (sammenlagt)

En aktivitet er en fællesnævner for mailingliste og seminar, da de to ikke er adskilt i databasen, og det er ikke muligt at skelne i MAOV-systemet.

3.3.9 Kontakt-person (nyt navn)

Kontakt-person er en kontaktperson som der er hos kunden. Navnet er ændret fra kontaktliste, da vi forventede at der var en liste for hver enkelt medarbejders kontaktpersoner hos de forskellige kunder. I stedet er de kun link fra kunden til samtlige af deres kontaktpersoner.

3.3.10 Pris-liste (ændret)

Prislisten står for sig selv, og linker ikke til noget andet. Der er kun priser på produkter, og ikke på konsulent-ydelser. Alle PtG-brugere har adgang til prislisten gennem MAOV-systemet.

3.3.11 De stiplede

- **Telefonliste (fjernet):** Er nedgraderet til en telefonnummer-attribut på medarbejderen.
- **Partnere (fjernet):** Partnere opfattes som kunder.
- **Produkt (fjernet):** Der er ikke nogen produkt-database med relevante oplysninger for MAOV-systemet.

- **Kalender:** Ikke tilgængelig. *(kommer måske som selvstændig PtG-service)*
- **Time-seddel:** Ikke tilgængelig.
- **Faglige ressourcer:** Ikke tilgængelig.
- **Opportunity:** Ikke tilgængelig.



3.4 Database-beskrivelse

MAOV-systemet skal tilgå en hel del af Oracle's eksisterende databaser, samt MAOV's egen database. I dette afsnit vil samtlige af Oracle's databaser, som MAOV-systemet tilgår, blive beskrevet. Beskrivelsen vil indeholde hvordan de enkelte databaser tilgås, hvilke tabeller og attributter der er interessante, og hvordan de hænger sammen. I det efterfølgende afsnit vil MAOV's egen database og tabeller blive beskrevet.

Inden gennemgangen skal det nævnes at der ikke er direkte adgang til hver enkelt af de forskellige af Oracle's databaser. Det foregår istedet via MAOV's egen database. Det vil sige at der i MAOV databasen er oprettet nogle db-links til de relevante databaser. Disse db-links giver kun læserettigheder, mens der er fuld læse/skrive-rettigheder til MAOV's egen database.

3.4.1 Oracle's databaser

De Oracle databaser og tabeller, som MAOV-systemet skal tilgå, er naturligvis bestemt af funktionsafgrænsningen. Som før nævnt, tilgår MAOV-systemet de enkelte databaser, via db-links som er oprettet i MAOV's egen database. Databaser der er oprettet db-links på er: prod, cdm og sms. I prod databasen er de relevante tabeller: hrv_people_info, ra_customers og product_license. I cdm databasen er de relevante tabeller: cdmcomp, cdmpcode, cdmcont og cdmacti. Og i sms databasen er de relevante tabeller: tar_head og tar_status.

De databaser og tabeller nævnt ovenfor har ikke været lige nemme at finde frem til. Det har krævet en hel del analyse arbejde, og snak frem og tilbage med Asger Jensen, Rasmus Menche og Bo Nielsen (se de relevante møde-referater i bilag 8.13 fra side 104).

For at oprette et db-link, skal der i den database man vil have adgang til, være oprettet en bruger man kan benytte. I vores tilfælde blev der oprettet en bruger i hver af de tre databaser, som det så var muligt at benytte. Når db-linket er oprettet, så har man præcis de samme rettigheder som den bruger man benytter. I vores tilfælde kun læserettigheder. Brugeren som db-linket benytter, kan yderligere have adgang til diverse skemaer i den specifikke database. Det var tilfældet med "prod" og "cdm" databaserne.

Måden man får adgang til en tabel på via et db-link foregår på følgende måde:

- skemanavn(hvis nødvendigt).tabelnavn@db-link-navn

Dvs. hvis man skal tilgå tabellen **ra_customers** i **prod** databasen, som ligger i skemaet **oradk**, så ser det således ud:

- **oradk.ra_customers@prod.dk.oracle.com**.

Herunder er en komplet oversigt over hvordan de forskellige tabeller tilgås, bemærk at der ikke er specificeret et skemanavn til sms databasen, da det ikke er nødvendigt.

- oradk.hrv_people_info@prod.dk.oracle.com
- oradk.ra_customers@prod.dk.oracle.com
- oradk.product_license@prod.dk.oracle.com
- cdm.cdmcomp@cdm.dk.oracle.com
- cdm.cdmpcode@cdm.dk.oracle.com
- cdm.cdmcont@cdm.dk.oracle.com
- cdm.cdmacti@cdm.dk.oracle.com
- tar_head@sms.dk.oracle.com
- tar_status@sms.dk.oracle.com



De enkelte databaser og tabeller beskrives i de følgende afsnit. Men inden denne gennemgang er der to tabeller, der kræver en særlig forklaring. Det drejer sig om "ra_customers" og "cdmcomp" fra hver deres database, som begge indeholder kundeoplysninger.

For at hente oplysninger om en kundes licenser og TAR's kræver det at man bruger kundelD'et fra "ra_customers" tabellen, og for at hente oplysninger om en kundes aktiviteter og kontakter kræver det at man bruger kundelD'et fra "cdmcomp" tabellen. Dette er heller ikke noget problem, da de to tabeller kan linkes sammen (ra_customers.customer_number = cdmcomp.oldoracleid), men det viste sig at det ikke var muligt at linke alle kunder sammen. Dette skyldes at Oracle for ikke så lang tid siden har flyttet deres kundedata over i nogle nye tabeller. Dvs. at kunder der bliver oprettet herefter ikke får et "oldoracleid" i "cdmcomp" tabellen, og dette gør at det ikke er muligt at få fat i begge ID'er, hvilket der er krævet for at vise alle oplysninger om en kunde.

Vi vælger tabellen "ra_customers", som indgang til kunden, dvs. når man søger en kunde så er det i denne tabel kunden bliver søgt. Dette gør at når "oldoracleid" i "cdmcomp" er tom, så er det kun muligt at få kundelD'et fra "ra_customers", hvilket som før nævnt, gør at man kun kan se licens og TAR oplysninger om kunden. Begrundelsen for dette valg, er at oplysninger om en kundes licenser og TAR's er dem der har størst værdi for brugerne, og de vil på denne måde altid være tilgængelige.

3.4.2 prod databasen

I denne database står oplysninger om de enkelte kunder og hvilke produkter de har licens til. Desuden står der også oplysninger om alle medarbejdere. De tre relevante tabeller og de attributter MAOV-systemet bruger er beskrevet nedenfor.

Tabellen "hrv_people_info" indeholder medarbejder oplysninger. Når en medarbejder skal findes ved hjælp af den indtastede søgestreg, så er det attributten "keystring", der sammenlignes med.

```
hrv_people_info
- person_id          (primær-nøgle)
- first_name
- middle_names
- last_name
- telephone_number_1 (hjemme)
- telephone_number_2 (mobil)
- work_telephone     (arbejde)
- email_address
- full_name
- keystring          (medarbejderens navn med store bogstaver)
```

Tabellen "ra_customers" indeholder kunde oplysninger. Når en kunde skal findes ved hjælp af den indtastede søgestreg, så er det attributten "attribute11", der sammenlignes med. Som før nævnt er denne tabel indgangen til kunden, og det er heri at kundens navn og status står. For at finde kundens ID til kontakt og aktivitets tabellerne, samt kundens adresse og telefonnummer, linkes attributten "oldoracleid" fra "cdmcomp" tabellen i "cdm" databasen med attributten "customer_number".

```
ra_customers
- customer_id        (primær-nøgle)
- customer_name
- attribute11        (kundens navn med store bogstaver)
- status             (kundens status, A=aktiv)
- customer_number    (link til cdm.cdmcomp.oldoracleid)
```



Tabellen "product_license" indeholder licens oplysninger. For at finde ud af hvilke produkter en kunde har licens til, linkes attributten "customer_id" fra "ra_customers" tabellen med attributten "customer_id".

```
product_license
- customer_id (link til ra_customers.customer_id)
- product_descr (hvilket produkt drejer det sig om)
- version (hvilken version er det)
- quantity (antallet af licenser til dette produkt)
- licens_type (må ikke være null, hvis null så er licensen
termineret)
- platform (hvilket OS-plattform det kører på)
```

3.4.3 cdm databasen

I denne database står igen oplysninger om de enkelte kunder og hvilke kontakter der er hos hvilken kunde. Endvidere er der også oplysninger om alle aktiviteter (seminare/nyhedsbreve) Oracle har for tiden, og hvilke aktiviteter den enkelte kunde er tilmeldt og har været tilmeldt. De 4 relevante tabeller og de attributter MAOV-systemet bruger er beskrevet nedenfor.

Tabellen "cdmcomp" indeholder kunde oplysninger. Dette er den tabel, som tabellen "ra_customers" fra "prod" databasen linker med. Heri er kundens ID, der skal bruges til kontakt og aktivitets tabellerne, samt oplysninger om kundens adresse og telefonnummer. Dette er dog, som før nævnt, kun muligt at få fat på hvis attributten "oldoracleid" ikke er tom.

```
cdmcomp
- id (primær-nøgle)
- addr1
- phone
- oldoracleid (link til prod.hrv_people_info.customer_number)
- primarypostcode (link til cdmrcode.id)
```

Tabellen "cdmrcode" indeholder postnummer/by oplysninger. For at få en kundes bynavn, så linkes attributten "primarypostcode" fra "cdmcomp" tabellen med attributten "id".

```
cdmrcode
- id (primær-nøgle)
- town
```

Tabellen "cdmcont" indeholder kontakt oplysninger. Alle de kontakter der er hos en kunde findes ved at linke attributten "id" fra "cdmcomp" tabellen med attributten "primcomp".

```
cdmcont
- id (primær-nøgle)
- firstn
- lastn
- title
- direct
- mobile
- email
- delu (må ikke være null)
- primcomp (link til cdmcomp.id)
```



Tabellen "cdmacti" indeholder aktivitets oplysninger. Aktiviteter som en kunde er tilmeldt og har været tilmeldt findes ved at linke attributten "id" fra "cdmcomp" tabellen med attributten "primcomp". Det skulle også være muligt at tilmelde kunder til fremtidige aktiviteter, men det var ikke muligt at finde en attribut, der indeholdt en fremtidig dato. Derfor vil denne funktion ikke blive mulig.

```
cdmacti
- head      (kort aktivitetsbeskrivelse, må ikke være null)
- startdate (hvilke dato den enkelte aktivitet er oprettet)
- primcomp  (link til cdmcomp.id)
```

3.4.4 sms databasen

I denne database står oplysninger om de enkelte kunders sagsforløb (TAR's). De to relevante tabeller og de attributter MAOV-systemet bruger er beskrevet nedenfor.

Tabellen "tar_head" indeholder oplysninger om de enkelte TARs. En vigtig oplysning om en TAR er om det er kunden eller Oracle der forventes at gøre noget ved sagen. Den oplysning står i "tar_status"-attributten, som et nummer. Er nummeret under 18, er TAR'en stadig direkte aktiv. En sag kan dog vise sig at være en fejl i programmet som skal løses i USA. Hvis en sag venter på det, er "bug_status"-attributten mindre end 39, og TAR'en skal også vises.

```
tar_head
- tar_no
- contact_first_name
- contact_name
- contact_phone
- change_date
- severity
- component
- subject
- tar_status (link til tar_status.status)
- bug_status (link til tar_status.status)
- org_id     (link til prod.ra_customers.customer_id)
```

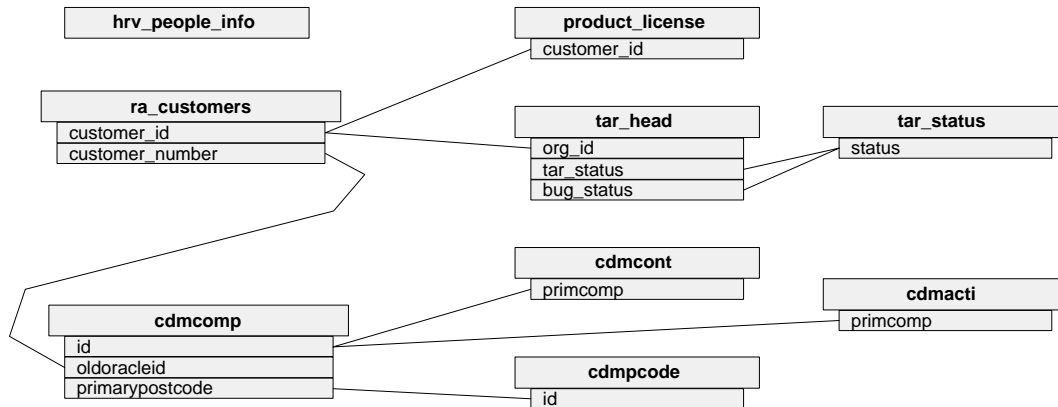
tar_status tabellen er til for at afkode de numre der står i "tar_status" og "bug_status" - attributterne i "tar_head"-tabellen. Man kan direkte linke tal-værdierne til denne tabel, og få en beskrivelse for det enkelte status-nummer, dog på engelsk.

```
tar_status
- status      (link til tar_head.tar_status og tar_head.bug_status)
- status_type ('T' = tar eller 'B' = bug)
- description
```



3.4.5 Overblik over link-attributter

Figuren herunder viser hvordan de forskellige tabeller linkes sammen.



3.4.6 Afrunding

Dette var en kort gennemgang af de forskellige Oracle databaser og tabeller, som MAOV-systemet anvender. De SELECT statements MAOV-systemet gør brug af kan ses i afsnit 4.3 "SQL-kommandoer" side 50.

Det var ærgerligt at det viste sig, at det ikke er muligt at få oplysninger om aktiviteter og kontakter for kunder, som er oprettet efter at kundedata er flyttet fra de gamle tabeller til de nye. Men det er der ikke noget at gøre ved, for sådan har Oracle selv valgt at skrue tabellerne sammen.

Dette gør selvfølgelig at MAOV-systemet ikke er 100% brugbart i alle situationer, men dette er Oracle helt indforstået med, da de selv har skabt dette problem. De vil dog efterfølgende forsøge at få lavet en linktabel, således at det vil være muligt at se alle oplysninger for samtlige kunder.

Det var desværre heller ikke muligt at finde nogle aktiviteter der ligger ude i fremtiden i aktivitets-tabellen. Dette bevirker at tilmelding til aktiviteter ikke bliver muligt, men det har ikke den store betydning. Det vigtigste er at kunne se hvilke aktiviteter en kunde er tilmeldt og har været tilmeldt. Det ville selvfølgelig have været smart, men det er en af de her "nice to have" funktioner, der sagtens kan undværes, hvilket Oracle også selv mener.



3.5 MAOV's egne tabeller

For at kunne holde styr på hvilke kunder og medarbejdere den enkelte bruger senest har brugt i systemet, og kunne vise priserne på alle Oracle's produkter, er det nødvendigt at oprette nogle tabeller i MAOV databasen, hvor disse oplysninger kan gemmes.

Om den enkelte bruger skal der kunne gemmes hvilke medarbejdere og kunder, brugeren senest har søgt og følgende valgt. I prislistens tilfælde så er det priserne for de enkelte produkter, samt hvilken produktgruppe produktet tilhører, som skal kunne gemmes. Udover dette så skal alle typer action-points også kunne gemmes, så det er nemt at slette/tilføje action-points. Hvor mange tabeller det drejer sig om, og hvordan de skal designes vil blive beskrevet i dette afsnit.

3.5.1 Senest søgte og valgte medarbejdere

Når en bruger har søgt og valgt en medarbejder, så skal medarbejderen gemmes i en tabel, sådan at når brugeren igen vælger menupunktet medarbejdere, så vil denne medarbejder blive listet op. Begrundelsen for denne funktionalitet, er at tidligere valgte medarbejdere typisk skal vælges igen. Det vil dog ikke være alle medarbejdere brugeren tidligere har valgt, som vil blive listet op, da dette meget hurtigt kan blive en lang liste. Men det vil dreje sig om de seneste 10 medarbejdere. Tallet 10 er vi kommet frem til ud fra en praktisk betragtning og ved snak med forskellige brugere, dvs. hvor mange personer er det egentlig relevant at gemme. Det tal vi landede på var 10, men det vil da være muligt at ændre dette tal senere hvis det bliver relevant.

De oplysninger der skal gemmes i en tabel for at få denne funktionalitet er: brugerens ID, medarbejderens ID og navn, samt et tidsstempel for hvornår medarbejderen blev valgt. Dette betyder at tabellen kommer til at bestå af 4 attributter. Attributten "userID" er brugerens ID, og den er samtidig primærnøgle, dvs. ud fra dette ID kan man finde de medarbejdere brugeren tidligere har valgt. Attributten "searchedForID" er medarbejderens ID, og skal bruges når der skal hentes oplysninger om denne medarbejder fra medarbejder tabellen. Attributten "searchedForName" er medarbejderens navn, og er det navn der kommer til at stå i listen. Attributten "timestamp" er det tidsstempel der bliver noteret når brugeren vælger en medarbejder, og bruges når de seneste medarbejdere skal findes frem.

Datatyperne for de enkelte attributter er ikke grebet ud af den blå luft, men er identiske med de tilsvarende attributter der eksisterer i Oracle's databaser. Dette skulle de også helst være, da der ellers kunne opstå problemer i bl.a en WHERE betingelse, f.eks hvis et ID blev til en varchar2, men oprindeligt er et number.

Vi vælger at kalde tabellen for "RecentEmpSearch". Herunder er tabellen vist med attributter og tilhørende datatyper.

```
RecentEmpSearch
- userID          varchar2(19)
- searchedForID   number(19)
- searchedForName varchar2(35)
- timestamp       date
```

3.5.2 Senest søgte og valgte kunder

Begrundelsen for at lave denne funktionalitet, er den samme som med medarbejdere, dvs. at tidligere valgte kunder typisk skal vælges igen. Det antal kunder som skal gemmes og listes op er det samme som for medarbejdere, altså 10. Og igen så kan dette tal senere ændres hvis det bliver relevant.



I denne tabel er der lidt flere oplysninger som skal gemmes, end der var for medarbejdertabellen. Det er fordi det tager noget længere tid at hente en kundes oplysninger op af databasen, og det kan derfor betale sig at gøre dette samtidig med søgningen. Når så den enkelte kunde vælges, så er oplysninger allerede hentet. Og det er disse oplysninger der også skal gemmes i tabellen. Derfor får tabellen hele 8 attributter, som er brugerens ID, kundens to ID'er (forskellige for hvilken Oracle database det drejer sig om), kundens stamoplysninger (navn, vej, by og telefonnummer) og et tidsstempel for hvornår kunden blev valgt.

Attributten "userID" er brugerens ID, og den er samtidig primærnøgle, dvs. ud fra dette ID kan man finde de kunder brugeren tidligere har valgt. Attributterne "searchedForID" og "searchedForCDMID" er kundens to ID'er, og skal bruges når der skal hentes diverse oplysninger om denne kunde fra Oracle's databaser. Attributterne "searchedForName", "custAddr1", "custTown" og "custPhone" er kundens stamoplysninger. Attributten "timestamp" er det tidsstempel der bliver noteret når brugeren vælger en kunde og bruges når de seneste kunder skal findes frem.

Datatyperne for de enkelte attributter er igen fuldt identiske med de tilsvarende attributter der eksisterer i Oracle's databaser.

Vi vælger at kalde tabellen for "RecentCustSearch". Herunder er tabellen vist med attributter og tilhørende datatyper.

```
RecentCustSearch
- userID          varchar2(19)
- searchedForID   number(19)
- searchedForCDMID varchar2(19)
- searchedForName varchar2(35)
- custAddr1       varchar2(35)
- custTown        varchar2(30)
- custPhone       varchar2(20)
- timestamp       date
```

3.5.3 Prislisten

Prislisten består af nogle produktgrupper, hvorunder de produkter og priser tilhørende denne produktgruppe står. Endvidere så skal der på forsiden af prislisten, der hvor produktgrupperne er listet op, stå som overskrift hvilken dato prislisten er fra. Udfra denne opbygning vælger vi at repræsentere prislisten ved tre tabeller: en til opstartsteksten, en til de enkelte produkter, og en til produktgrupperne.

Opstartsteksten: Denne tabel kalder vi for "pricelist_startuptext", og den får kun een attribut med navnet "startuptext", der er af datatypen varchar2 med en maksimal størrelse på 40 karakterer. Den vil kun komme til at indeholde een tupel, dvs. hver gang prislisten opdateres, så overskrives denne attribut med den nye dato for den aktuelle prisliste.

De enkelte produkter: Denne tabel kalder vi for "pricelist_details", og den kommer til at indeholde 6 attributter. Et produktgruppelID, et produktnavn og 4 priser. Attributten "id" er produktets gruppelID, dvs. den er fremmednøgle til produktgruppetabellen, og er af datatypen number. Attributten "productname" er navnet på produktet, og er af datatypen varchar2 med en maksimal størrelse på 50 karakterer. Attributterne "nuss", "nums", "upu" og "license" er produktets forskellige priser, og de er alle af datatypen number.

Produktgrupperne: Denne tabel kalder vi for "pricelist_groups", og den kommer til at indeholde to attributter. Et ID for hver produktgruppe og et produktgruppenavn. Attributten "id" er produktgruppens ID, den er primærnøgle og er af datatypen number. Attributten "groupname" er navnet på produktgruppen, og er af datatypen varchar2 med en maksimal størrelse på 30 karakterer.



Herunder er de tre tabeller vist med attributter og tilhørende datatyper.

```
pricelist_startuptext
- startuptext varchar2(40)
```

```
pricelist_details
- id          number          not null (pricelist_groups.id)
- productname varchar2(50) not null
- nuss        number          (Named User Single Server)
- nums        number          (Named User Multi Server)
- upu         number          (Universal Power Unit)
- license     number          (Price for one license)
```

```
pricelist_groups
- id          number          not null
- groupname   varchar2(30) not null
```

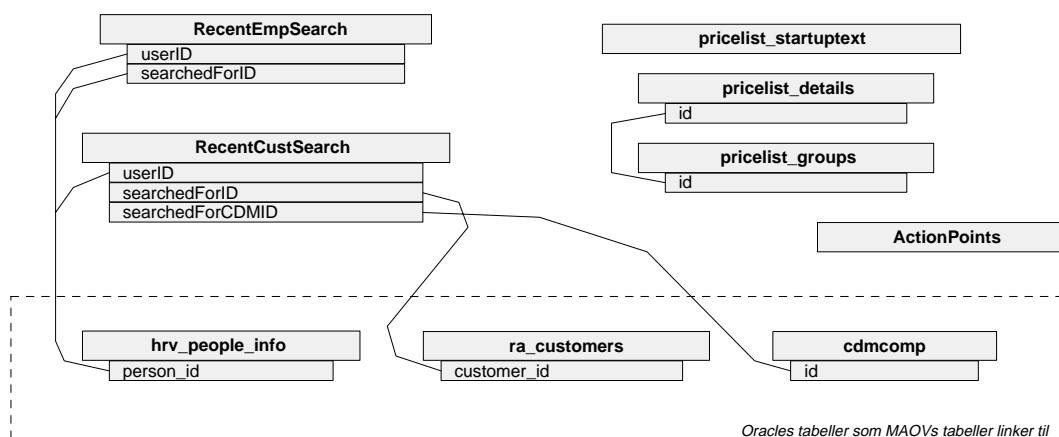
3.5.4 Action-points

De action-points, som det skal være muligt at sende, skal hentes fra en tabel. Begrundelsen for at lægge dem i en tabel, er at man nemt kan slette/tilføje action-points, samt redigere dem. Hvis de istedet havde været hardkodet, så ville dette være meget besværligt at gøre, da dette kræver at man skal direkte ind i koden og ændre det, hvilket ikke er hensigtsmæssigt.

Et action-point består af en overskrift og en beskrivelse, og vil blive sendt som en e-mail, det er derfor naturligt at tabellen får to attributter. Vi vælger meget nærliggende at kalde tabellen "ActionPoints", og attributterne for "subject" og "body". Attributten "subject" bliver en varchar2 på maksimalt 30 karakterer, dette skulle give plads nok til at skrive en sigende overskrift og den vil samtidig heller ikke fylde for meget af WAP-telefonens display (dvs. maks 2 linjer). Attributten "body" bliver en varchar2 på maksimalt 250 karakterer, dette skulle være nok til en kortfattet beskrivelse. Herunder er tabellen vist med attributter og tilhørende datatyper.

```
ActionPoints
- subject varchar2(30)
- body    varchar2(250)
```

Figuren herunder viser hvordan de forskellige tabeller linkes sammen.



3.5.5 Afrunding

Denne gennemgang skulle meget gerne have givet et indblik i hvordan MAOV's egne tabeller er tænkt designet og begrundelsen herfor.



3.6 Sekvens-diagrammer

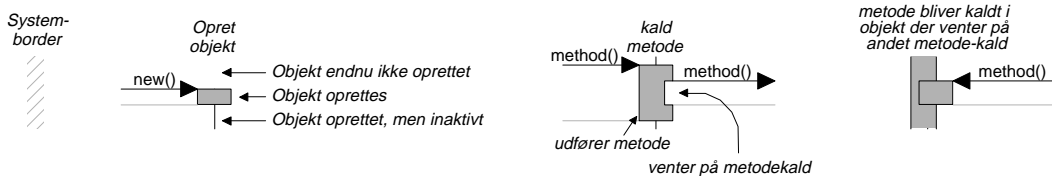
Måden objekterne kommunikerer med hinanden afviger ikke meget i de forskellige use-cases. Derfor er her nogle repræsentative eksempler på hvordan det foregår.

3.6.1 Notation

Vi har tilpasset måden sekvensdiagrammer tegnes på lidt, for bedre at kunne illustrere hvordan MAOV-systemet kommunikerer internt imellem sine objekter. Vi følger dog de grundlæggende regler. Hvert objekt har sin egen lodrette linje, men navn i toppen, og stimuli er vandrette linjer med pile der viser hvem der kalder hvem. De såkaldte "system-borders" er eksterne brugere af systemet der kommunikerer direkte.

Det er ikke alle objekterne der er oprettet hele tiden, så derfor vil vi gerne vise hvornår de forskellige objekter bliver oprettet. Når et objekt ikke er oprettet vises den tilhørende lodrette linje ikke.

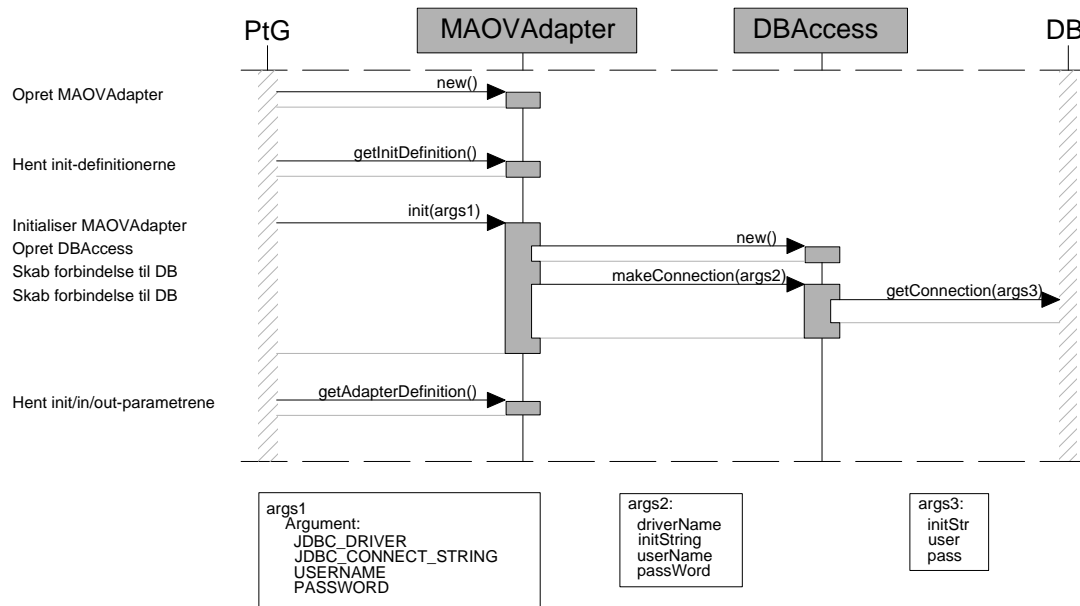
MAOV-systemet er implementeret i Java uden brug af tråde (multi-threading) internt i systemet, så der er kun eet objekt der kan udføre noget ad gangen. Når et objekt kalder et andet, venter det på at det andet returnere. Vi vil derfor gerne vise hvem det venter på hvem. Selvom et objekt venter, kan andre objekter dog stadig kalde andre metoder i det ventende objekt, så også det skal kunne vises.



Der er ikke plads til at skrive alle argumenterne på stimuli-linierne, så derfor er alle værdierne vist i kasser under sekvensdiagrammet med et navn (eks. `arg1`) der henviser til hvilke argumenter der hører til hvilket kald.



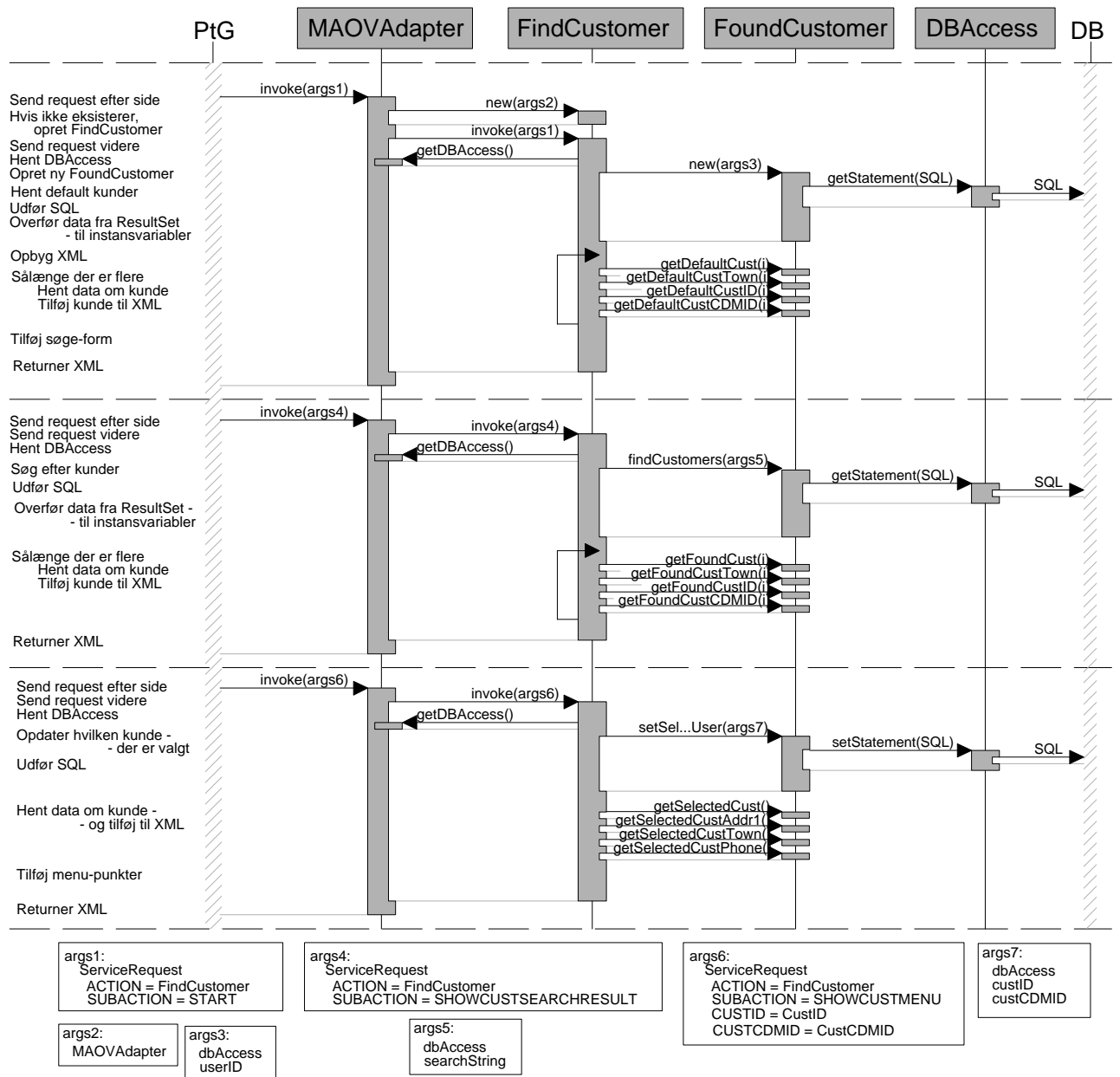
3.6.2 Oprettelse af MAOVAdapter



Dette er ikke en use-case, men det er relevant at vise hvordan MAOVAdapteren's eneste reelle bruger (PtG) kommunikerer med MAOVAdapteren, og ikke mindst hvordan den opretter en ny Adapter. Det er nemlig ikke bare at køre en constructor (hvad det jo heller ikke er i en Applet).



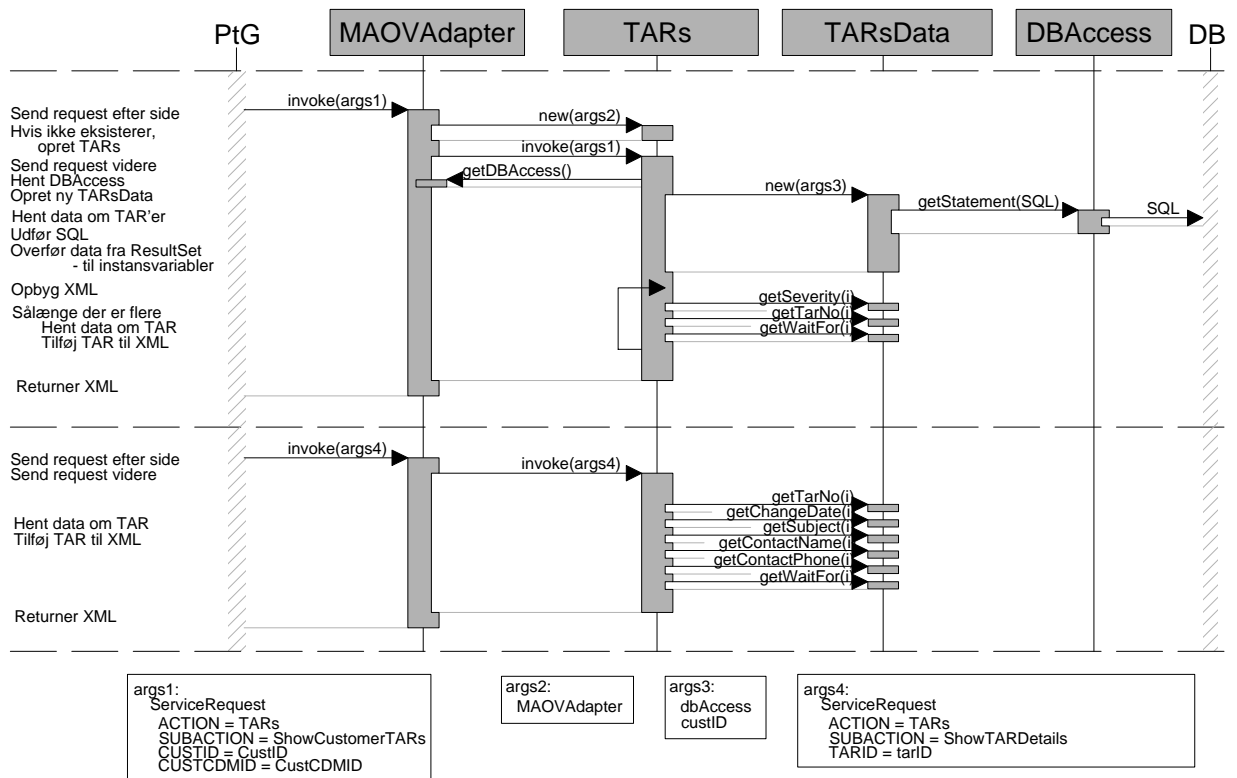
3.6.3 Søg kundeoplysninger



Denne use-case starter før brugeren har fundet en kunde, og ender med at brugeren har fundet en kunde, som man så kan manøvrere videre i systemet med. Der er tre faser, der hver svarer til at systemet genererer en side. Den midterste kan springes over, idet det er den side der genereres når en bruger vil søge efter en kunde, og har indtastet en søge-streng. Man kan altså hoppe direkte fra den første fase til den tredje fase, eller gå alle tre igennem.



3.6.4 Se oplysninger om aktuelle sager.



For at komme hertil skal man have fundet en kunde. Derefter hentes data om TARs fra databasen, men kun for den kunde. Når brugeren senere vil se flere detaljer om en enkelt TAR (anden fase), er data allerede hentet fra databasen, og hentes derfor bare fra entitetsobjektet (TARsData). Hvis brugeren vælger en anden kunde og vil se TAR's for den, oprettes et nyt TARsData-objekt, og det gamle destrueres af garbage-collectoren.

3.7 Konklusion

Der er foretaget en række designovervejelser, som MAOV-systemet er blevet designet og implementeret efter. Desuden er de databaser, som skal tilgås blevet beskrevet og måden hvorpå objekterne kommunikerer sammen er specificeret.

Under undersøgelserne af Oracle's databaser viste det sig, at der ikke var tilstrækkelige data til at implementere MAOV-systemet ud fra den oprindelige domæne model. Det var derfor nødvendigt at tilrette domæne modellen, for at få et korrekt overblik over hvordan data hænger sammen.

Design-detallerne er udarbejdet sideløbende med implementeringen af de enkelte funktioner og har vist sig at være meget effektivt.



4. Implementering

4.0.1 Indledning

I dette kapitel vil forskellige aspekter af implementeringen blive beskrevet. Læseren forventes at vide generelt hvordan MAOV-systemet er designet, og have en generel indsigt i Java's håndtering af klasser, objekter og forskellen på public og private.

Selve source-koden er ikke inkluderet i dette afsnit, men der er beskrivelser af forskellige relevante områder vedrørende koden. Der er et klassediagram der viser de forskellige klasser, og deres kendskabs-relationer. Der er en grundlæggende forklaring af hvordan program-koden er bygget op, så man nemt kan få et overblik over koden. Der er de komplette SQL-kommandoer, og en forklaring af hvordan systemet husker hvilke kunder/medarbejdere brugeren sidst har haft valgt. Desuden er en komplet liste over de input-parametre systemet modtager, og en forklaring til hvordan man kan ændre systemet. Endelig er der en liste over forbedringer der kan tilføjes systemet senere.

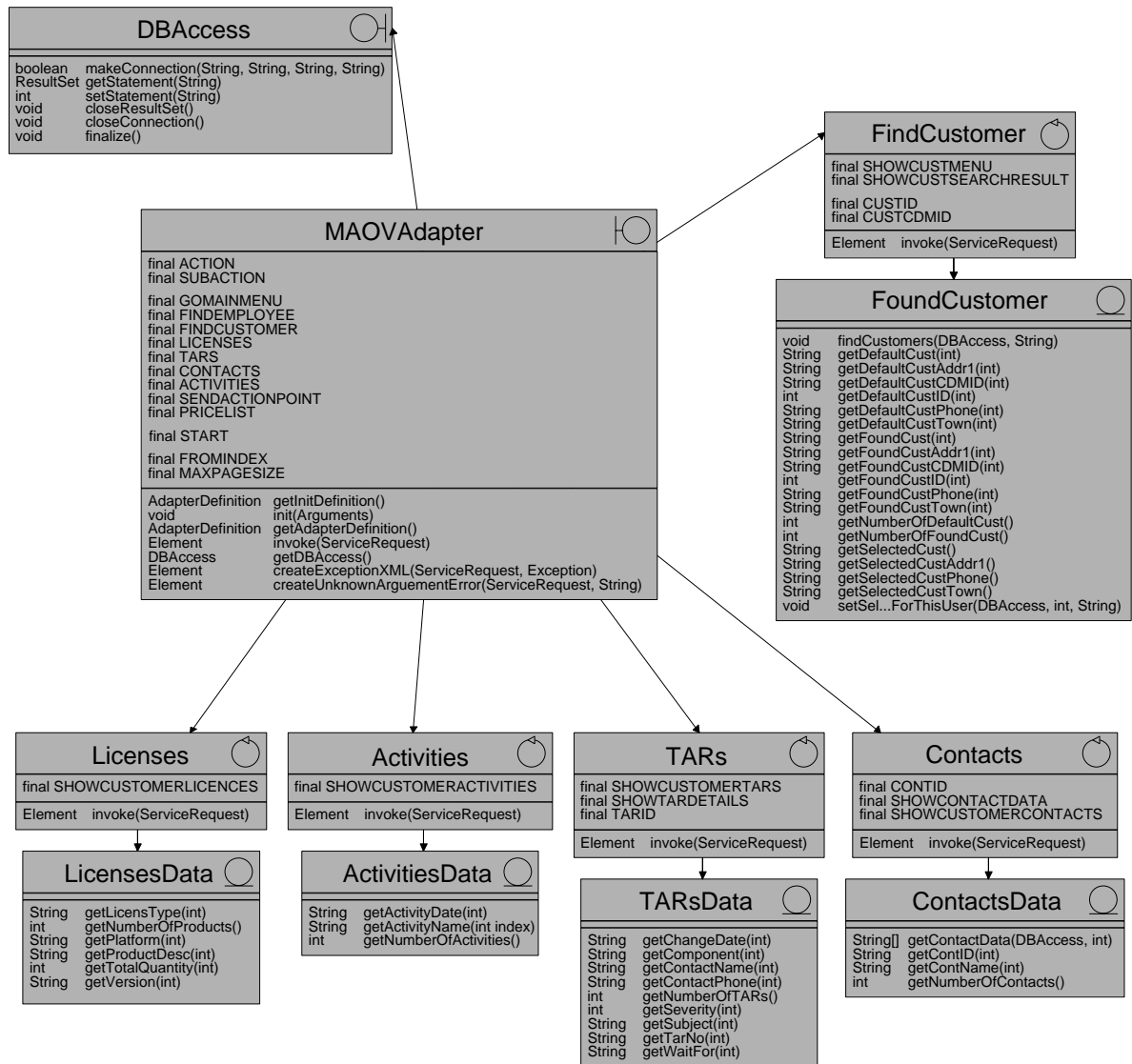
Source-kode, det kompilerede program, JavaDoc'en og diverse forsøgs-programmer er vedlagt på digital form på CD'en der er vedlagt projektet som bilag.



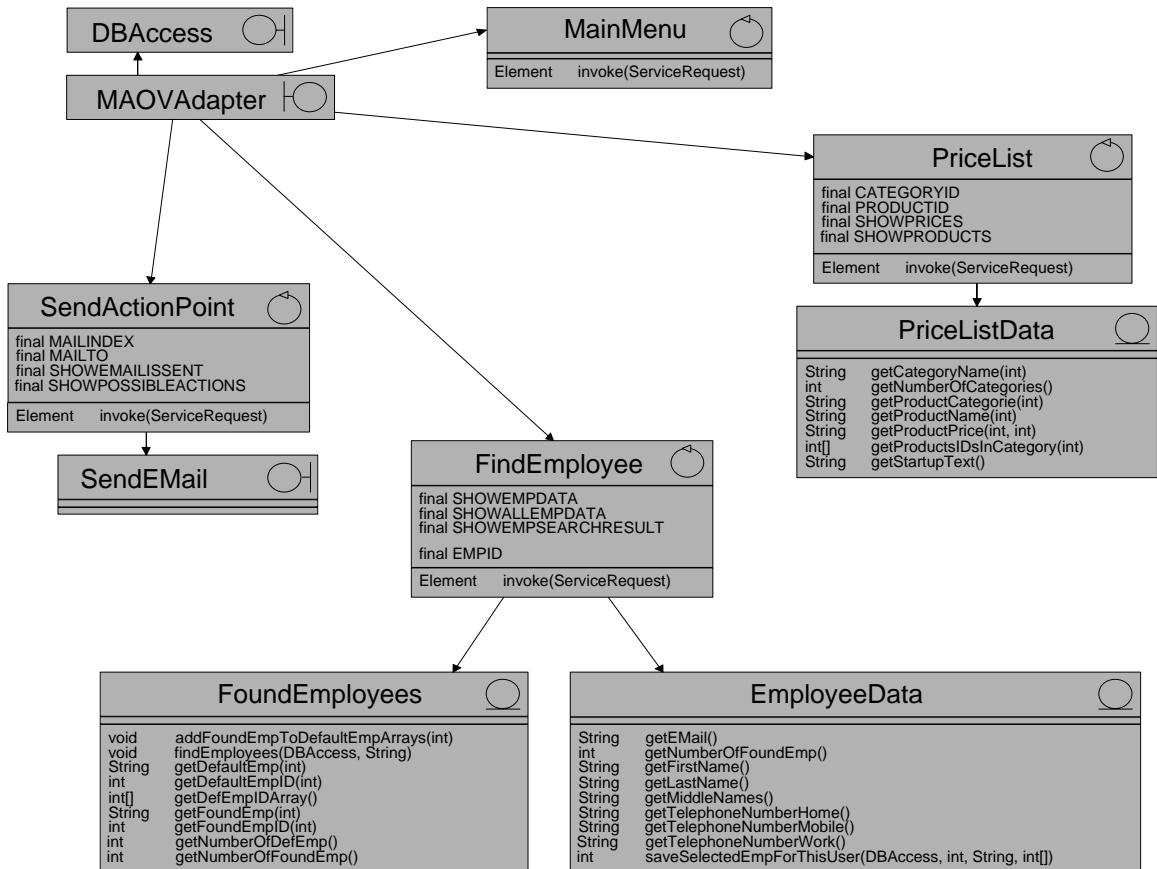
4.1 Klassediagram

Klassediagrammet viser de forskellige klasser og hvilke instans-variabler og metoder der er public. Klasse-navnene og kendskabsrelationerne er de sammen som i ICE-modellen.

Af tegne-tekniske årsager er klasse-diagrammet delt ud på to illustrationer, hvor den første indeholder alt hvad der har med kunder at gøre.



Den næste illustration indeholder resten af systemet, som der ikke var plads til på den første tegning. Bemærk at MAOVAdapter'en og DBAccess bruges på begge illustrationer, men at deres indhold kun er beskrevet på den første.



De mange instans-variabler er værdier for URL-argumenterne. Det er ikke dem alle der bliver brugt af andre klasser, selvom de er public. Man kan derfor sætte mange af dem private. At de alligevel er public, skyldes at systemet holdes åbent for at linke på tværs af funktionerne

Læg mærke til at alle de klasser der skal generere XML kun indeholder een enkelt metode hver, nemlig invoke(ServiceRequest). Det er altså den metode der genererer samtlige af de forskellige sider, som denne klasse kan generere. I koden har hver enkelt side dog fået sin egen metode, men eftersom det kun er invoke(ServiceRequest) der må kaldes udefra er de andre sat til private.

SendEmail ser lidt tom ud, idet der hverken er instans-variabler eller metoder. Det hænger sammen med at alt hvad den gør, foregår i constructor'en, og når den er færdig, smides objektet væk.

De forskellige entitets-objekter indeholder typisk kun get-metoder, og man kan undre sig over hvor de får data fra. De henter data fra databasen i constructoren som ikke er vist på klassediagrammet.



4.2 Manøvrering i koden

De endelige klasser som de er implementeret, er de samme som er illustreret på ICE-modellen og klasse-diagrammet. Herunder er en mere forklarende beskrivelse af hvordan systemet fungerer.

4.2.1 MAOVAdapter

Hovedklassen er MAOVAdapter, og det er den der er selve interfacet til Portal-to-Go. Alle forespørgelser går gennem dette objekt. MAOVAdapteren gør dog ikke noget ved dem, andet end at distribuere dem videre til de klasser der kan tage sig af dem.

MAOVAdapteren implementerer en oracle.panama.adapter.Adapter, så Portal-to-Go kan genkende og bruge den som Adapter. MAOVAdapteren skal derfor implementere følgende fire metoder:

- **getInitDefinition()** - Adapteren fortæller PtG hvilke init-parametre den accepterer, og eventuelt deres default-værdier.
- **init()** - Adapteren initialiseres. PtG kan således bruge andet end default-værdierne.
- **getAdapterDefinition()** - Adapteren fortæller PtG hvilke init/input/output-parametre den er initialiseret med.
- **invoke()** - PtG forespørger Adapteren efter en side.

4.2.2 invoke(serviceRequest)

Det er denne metode Portal-to-Go kalder hver gang brugeren beder om en ny side. Den får en "oracle.panama.ServiceRequest" med som argument, og ud fra den udledes hvilken side der er ønsket. Det gøres ved at hente parametrene ud. MAOV-systemet har to grundlæggende parametre: ACTION og SUBACTION. ACTION definerer hvilken klasse der skal tage sig af forespørgselen. Hver klasse har dog forskellige opgaver så SUBACTION fortæller hvilken underopgave det er, klassen skal tage sig af. MAOVAdapteren kigger ikke selv på SUBACTION.

De klasser der får requestene fra MAOVAdapteren er defineret som kontrol-objekter på ICE-modellen. De opfører sig nogenlunde ens, så derfor vil de her blive beskrevet generelt.

4.2.3 Kontrol-objekterne

De har hver en invoke-metode som MAOVAdapteren kalder. I den undersøges SUBACTION og den dertil hørende metode kaldes. I den opbygges den side der skal præsenteres som XML, med hjælpeklassen "oracle.panama.PAPrimitive". Det færdige XML-dokument returneres til MAOVAdapteren som igen returneres til Portal-to-Go, der endeligt sørger for at siden vises korrekt.

Et menupunkt på en side vil altid være et link til den samme MAOVAdapter, men med forskellige parametre, som så præciserer hvilken side der denne gang ønskes. Der vil typisk være tilføjet flere parametre end ACTION og SUBACTION. Således vil eksempelvis kunde-id også være sat på som en parameter.

De sider der genereres indeholder typisk oplysninger hentet fra Oracles databaser. Hele tilgangen til databaserne er dog adskilt fra den XML-genererende del af systemet, og gemt væk i entitets-objekterne, som de er illustreret på ICE-modellen. Alle disse entitets-objekter er generelt ens ligesom kontrol-objekterne, så de vil også blive beskrevet generelt her.



4.2.4 Entitets-objekterne

Entitets-objekterne henter selv deres relevante oplysninger fra databasen når de oprettes. Det sker altså i constructor'en. De deler alle et objekt af klassen DBAccess, der hjælper med at udføre SQL-forespørgsler. Alle data bliver derefter gemt i instans-variabler, som kan tilgås med get-metoder. På den måde er database-tilgangen gemt af vejen, og når først entitets-objektet er oprettet, er der hurtig adgang til alle data'ene.

Når der er mange data i et entitets-objekt gemmes de i array's, da det er den mest direkte måde at gemme data på. Men array's har fast størrelse, og de ResultSet vi får ud af databasen har ukendt størrelse. Derfor tømmes ResultSet'et først ud i en Vector, og udfra Vector'ens længde oprettes arrays'ene og data flyttes fra Vectoren til arrays'ene. Det kan virke omstændigt, men det er smartere end at spørge databasen med "SELECT COUNT(*) FROM ..." om antallet af tupler.

Det er ikke alle oplysninger der kan hentes frem under oprettelsen af entitets-objekterne. Derfor har nogle af dem yderligere metoder der henter oplysninger ud over det der blev hentet i constructoren. I de tilfælde hvor der skal gemmes data i databaserne, er selve database-håndteringen også lagt i entitets-objekterne.

Dette burde give et overblik over systemet. Der er mere detaljerede oplysninger om klasserne og metoderne i javadoc'en (vedlagt på CD).

4.3 SQL-kommandoer

I det følgende er alle de SQL kommandoer, som vi har brugt listet op. Først vil alle SELECT statements blive vist, og derefter alle CREATE statements for MAOV's egne tabeller med tilhørende INSERT statements hvor relevant. Gennemgangen af de forskellige SELECT statements foregår på den måde, at de er inddelt efter hvilket objekt de bruges i, samt at der oven over det enkelte SELECT statement vil være en lille beskrivelse af hvad det udfører. For de forskellige CREATE statements vil gennemgangen foregå på den måde, at de vil være inddelt efter hvilken funktion den oprettede tabel eller de oprettede tabeller understøtter, og udover dette vil der, hvor relevant, være vist INSERT statements.

Alle de SELECT statements vi har brugt er listet op i det følgende.

4.3.1 FoundCustomers

Henter de 10 seneste søgte kunder for en bruger.

```
SELECT
  searchedForName,
  searchedForID,
  searchedForCDMID,
  custAddr1,
  custTown,
  custPhone
FROM
  RecentCustSearch
WHERE
  userID = 'userID'
  AND rownum < 11
ORDER BY
  timeStamp DESC
```

Henter kunders ID'er og stamdata udfra en søgestreg indtastet af brugeren i søgeform.



```
SELECT
  cprod.customer_name,
  cprod.customer_id,
  NVL(ccdm.id, 'null'),
  NVL(ccdm.addr1, 'null'),
  NVL(pcode.town, 'null'),
  NVL(ccdm.phone, 'null')
FROM
  oradk.ra_customers@prod.dk.oracle.com cprod,
  cdm.cdmcomp@cdm.dk.oracle.com ccdm,
  cdm.cdmpcode@cdm.dk.oracle.com pcode
WHERE
  cprod.attributell like '% searchString.toUpperCase() %'
  AND cprod.status = 'A'
  AND cprod.customer_number = ccdm.oldoracleid (+)
  AND ccdm.primarypostcode = pcode.id (+)
ORDER BY
  cprod.customer_name
```

4.3.2 FoundEmployees

Henter de 10 seneste søgte medarbejdere for en bruger.

```
SELECT
  searchedForName,
  searchedForID
FROM
  RecentEmpSearch
WHERE
  userID = 'userID'
  AND rownum < 11
ORDER BY
  timeStamp DESC
```

Henter medarbejderes ID og navn ud fra en søgestreg indtastet af brugeren i søgeform.

```
SELECT
  person_id,
  full_name
FROM
  oradk.hrv_people_info@prod.dk.oracle.com
WHERE
  keystring like '% searchString %'
ORDER BY
  full_name
```

4.3.3 EmployeeData

Henter stamdata om en valgt medarbejder.

```
SELECT
  first_name,
  middle_names,
  last_name,
  telephone_number_1,
  telephone_number_2,
  work_telephone,
  email_address,
  full_name
FROM
  oradk.hrv_people_info@prod.dk.oracle.com
WHERE
  person_id = empID
```



4.3.4 LicensesData

Henter oplysninger om alle de produkter en kunde har licens til.

```
SELECT
  product_descr,
  version,
  sum(nvl(quantity,0)) total_quantity,
  licens_type
FROM
  oradk.product_license@prod.dk.oracle.com
WHERE
  customer_id = custID
  AND licens_type IS NOT NULL
GROUP BY
  product_descr,
  version,
  licens_type
```

4.3.5 ActivitiesData

Henter alle de aktiviteter en kunde har været og er tilmeldt til.

```
SELECT DISTINCT
  nvl(acti.head,'null'),
  acti.startdate
FROM
  cdm.cdmcomp@cdm.dk.oracle.com comp,
  cdm.cdmacti@cdm.dk.oracle.com acti
WHERE
  acti.head is not null and
  comp.id = acti.primcomp and
  comp.id = 'companyID'
ORDER BY
  startdate desc
```

4.3.6 TARsData

Henter oplysninger om alle de TAR's der er aktive for en kunde.

```
SELECT
  a.tar_no,
  a.contact_first_name,
  a.contact_name,
  a.contact_phone,
  a.change_date,
  a.severity,
  a.component,
  a.subject,
  a.tar_status,
  a.bug_status,
  b.description,
  c.description
FROM
  tar_head@sms.dk.oracle.com a,
  tar_status@sms.dk.oracle.com b,
  tar_status@sms.dk.oracle.com c
WHERE
  b.status = a.tar_status AND
  b.status_type = 'T' AND
  c.status(+) = a.bug_status AND
  c.status_type(+) = 'B' AND
  (a.tar_status<18 OR NVL(a.bug_status, 39)<39) AND
  a.org_id = 'custID'
ORDER BY
  a.severity,
  a.change_date
```



4.3.7 ContactsData

Henter alle de kontakter der er hos en kunde.

```
SELECT DISTINCT
  cn.firstn,
  cn.lastn,
  cn.title,
  cn.direct,
  cn.mobile,
  cn.email
FROM
  cdm.cdmcont@cdm.dk.oracle.com cn
WHERE
  cn.delu is null and
  cn.id = 'contID'
```

Alle de CREATE statements vi har brugt er listet op i det følgende, med tilhørende INSERT statements, hvor relevant.

4.3.8 Senest søgte og valgte kunder

```
create table RecentCustSearch (
  userID varchar2(19),
  searchedForID number(19),
  searchedForCDMID varchar2(19),
  searchedForName varchar2(35),
  custAddr1 varchar2(35),
  custTown varchar2(30),
  custPhone varchar2(20),
  timestamp date
);
```

4.3.9 Senest søgte og valgte medarbejdere

```
create table RecentEmpSearch (
  userID varchar2(19),
  searchedForID number(19),
  searchedForName varchar2(35),
  timestamp date
);
```

4.3.10 Prislisten

```
create table pricelist_startuptext (
  startuptext varchar2(40)
);

create table pricelist_details (
  id number not null,
  productname varchar2(50) not null,
  nuss number, /* Named User Single Server */
  nums number, /* Named User Multi Server */
  upu number, /* Universal Power Unit */
  license number
);

create table pricelist_groups (
  id number not null,
  groupname varchar(30) not null
);

/* Startuptext */
insert into pricelist_startuptext values ('E-Business Global Price List 13.10.00');

/* Database */
insert into pricelist_groups values (1,'Oracle Database');
```



```
insert into pricelist_details values(1,'Oracle Database Standard
Edition',1299,1624,122,null);
insert into pricelist_details values(1,'Oracle Database Enterprise
Edition',4872,6090,812,null);
insert into pricelist_details values(1,'Trusted Oracle Enterprise
Edition',5847,7308,974,null);
insert into pricelist_details values(1,'Oracle Database Personal
Edition',3208,null,null,null);
insert into pricelist_details values(1,'Oracle Database Lite',2396,null,null,null);
insert into pricelist_details values(1,'Advanced Security',974,1218,122,null);
insert into pricelist_details values(1,'Parallel Server',1949,2436,244,null);
insert into pricelist_details values(1,'Partitioning',1949,2436,244,null);
insert into pricelist_details values(1,'Spatial',2923,3654,325,null);
insert into pricelist_details values(1,'Diagnostic Management Pack',325,406,81,null);
insert into pricelist_details values(1,'Tuning Management Pack',325,406,81,null);
insert into pricelist_details values(1,'Change Management Pack',325,406,81,null);
insert into pricelist_details values(1,'Management Pack for SAP R/3',325,406,81,null);

/* Application Server */
insert into pricelist_groups values (2,'Application Server');
insert into pricelist_details values(2,'Internet Application Server Standard
Edition',284,365,41,null);
insert into pricelist_details values(2,'Internet Application Server Enterprise
Edition',1624,2030,244,null);
insert into pricelist_details values(2,'Internet Application Server Wireless
Edition',771,974,1218,null);

/* Tools */
insert into pricelist_groups values (3,'Tools');
insert into pricelist_details values(3,'Internet Developer
Suite',32441,40562,null,null);
insert into pricelist_details values(3,'Discoverer Plus',6456,8080,null,null);
insert into pricelist_details values(3,'SQL*Plus',4020,5035,null,null);
insert into pricelist_details values(3,'Programmer',8080,10110,null,null);

/* Integration Products */
insert into pricelist_groups values (4,'Integration Products');
insert into pricelist_details values(4,'Open System Gateways',null,null,null,121807);
insert into pricelist_details values(4,'Mainframe Integration
Gateways',null,null,null,771443);
insert into pricelist_details values(4,'Enterprise Integration
Gateways',null,null,null,771443);
insert into pricelist_details values(4,'EDA/SQL Gateways',974454,null,null,null);
insert into pricelist_details values(4,'Each Additional EDA/SQL
Driver',null,null,null,487227);
insert into pricelist_details values(4,'Applications InterConnect
Toolkit',812,1015,122,null);

/* Other Server Products */
insert into pricelist_groups values (5,'Other Server Products');
insert into pricelist_details values(5,'Video Server',325,406,81,null);
insert into pricelist_details values(5,'Email Server',122,154,81,null);
insert into pricelist_details values(5,'Internet Directory',null,null,406,41);
insert into pricelist_details values(5,'Message Broker',771,974,81,null);

/* Data Warehousing Products */
insert into pricelist_groups values (6,'Data Warehousing Products');
insert into pricelist_details values(6,'Warehouse Builder',null,null,203,null);
insert into pricelist_details values(6,'Pure Name & Address (US)',null,null,244,null);
insert into pricelist_details values(6,'Pure Name & Address
(Canada)',null,null,65,null);
insert into pricelist_details values(6,'Geocode',null,null,41,null);
insert into pricelist_details values(6,'Oracle Data Mining Suite',null,null,406,null);
insert into pricelist_details values(6,'Express Server',12952,16200,1624,null);
insert into pricelist_details values(6,'Express Analyzer',4832,6050,null,null);
insert into pricelist_details values(6,'Express Objects',32441,40562,null,null);
```

4.3.11 Action points

```
create table ActionPoints (
  subject varchar2(30),
  body varchar2(250)
);

insert into table ActionPoints values('Ring','Ring hurtigst muligt til mig.');
```



4.4 Hvordan gemmes valgte kunder og medarbejdere

Når en bruger vælger en kunde eller en medarbejder, så skal det registreres i henholdsvis "RecentCustSearch" tabellen og "RecentEmpSearch" tabellen. Det er dog ikke lige meget hvordan dette foregår. Hvis det drejer sig om en ny kunde eller medarbejder, dvs. det er første gang brugeren vælger en af de to, så skal der indsættes en ny tuppel i den tabel det drejer sig om. Hvis brugeren istedet vælger en kunde eller medarbejder, som før har været valgt, så vil der allerede være en tuppel i en af de respektive tabeller, hvilket betyder at tuplen skal opdateres.

I det følgende er måden hvorpå de valgte kunder og medarbejdere gemmes på beskrevet. Det foregår i to klasser, en for kunder og en for medarbejdere. En metode i hver af disse klasser, står for denne funktionalitet, og det er dem der beskrives i det følgende. Ønskes yderligere dokumentation af disse klasser, se JavaDoc på vedlagte CD.

Kunder beskrives først og derefter medarbejdere.

4.4.1 Kunder (FoundCustomers)

Når en bruger vælger en kunde, så skal den valgte kunde indsættes (hvis valgt for første gang) eller opdateres (hvis valgt før) i "RecentCustSearch" tabellen. Dette foregår ved at kalde metoden "setSelectedCustDataAndSaveSelectedCustForThisUser" i entitets-objektet "FoundCustomers". Metoden tager tre argumenter: en reference til DBAccess, den valgte kundes ID og brugerens ID.

Måden hvorpå det bestemmes om brugeren har valgt kunden for første gang, eller om kunden har været valgt før, er ved at løbe "defaultCustID" array'et igennem. Hvis den valgte kundes ID findes heri betyder det at brugeren har haft valgt kunden før. Det er dog, som tidligere nævnt, kun de 10 seneste valgte kunder det drejer sig om. Nedenstående kode viser hvordan dette bestemmes.

```
int defaultCustIndex = 0;
boolean update = false;
for (int i=0;i<getNumberOfDefaultCust();i++) {
    if (defaultCustID[i] == custID) {
        defaultCustIndex = i;
        update = true;
    }
}
```

Hvis variabelen "update" er false, betyder det at den valgte kunde ikke er at finde blandt de seneste valgte kunder, og derfor skal der indsættes en ny tuppel i tabellen. De data der skal indsættes findes ved at løbe "foundCustID" array'et igennem indtil man finder den valgte kunde (kundeID). Nu har man et indeks til den valgte kunde, og det bruges i den følgende INSERT statement. Efter at kunden er indsat i tabellen, skal kunden også indsættes i de arrays der indeholder de seneste valgte kunder (kun vist metodekaldet), hvis dette ikke gøres kan man risikere at en kunde kan blive indsat to gange. Nedenstående kode viser hvordan dette foregår.

```
String sql;
if (!update) {
    int foundCustIndex = 0;
    for (int i=0;i<getNumberOfFoundCust();i++) {
        if (foundCustID[i] == custID) foundCustIndex = i;
    }
    sql = "INSERT INTO RecentCustSearch values( "+
        ""+ userID +"', "+
```



```
" "+ custID +", "+
"'"+ foundCustCDMID[foundCustIndex] +"' , "+
"'"+ foundCust[foundCustIndex] +"' , "+
"'"+ foundCustAddr1[foundCustIndex] +"' , "+
"'"+ foundCustTown[foundCustIndex] +"' , "+
"'"+ foundCustPhone[foundCustIndex] +"' , "+
"sysdate )";
```

```
addFoundCustToDefaultCustArrays(foundCustIndex);
}
```

Hvis variabelen "update" er true, betyder det at den valgte kunde har været valgt før, og derfor skal tabellen opdateres. Nedenstående kode viser hvordan dette foregår.

```
else {
    sql = "UPDATE RecentCustSearch "+
        "SET timeStamp = sysdate "+
        "WHERE userID = '"+ userID +"' "+
        "AND searchedForID = '"+ custID;
}
```

SQL-sætningen udføres ved at benytte DBAccess referencen.

```
dbAccess.setStatement(sql);
```

4.4.2 Medarbejdere (EmployeeData)

Når en bruger vælger en medarbejder, så skal den valgte medarbejder indsættes (hvis valgt for første gang) eller opdateres (hvis valgt før) i "RecentEmpSearch" tabellen. Dette foregår ved at kalde metoden "saveSelectedEmpForThisUser" i entitets-objektet "EmployeeData". Metoden tager fire argumenter: en reference til DBAccess, den valgte medarbejders ID, brugerens ID og en reference til "defaultEmpID" array'et fra entitets-objektet "FoundEmployees".

Måden hvorpå det bestemmes om brugeren har valgt medarbejderen for første gang, eller om medarbejderen har været valgt før, er ved at løbe "defaultEmpID" array'et igennem. Hvis den valgte medarbejders ID findes heri betyder det at brugeren har haft valgt medarbejderen før. Det er dog, som tidligere nævnt, kun de 10 seneste valgte medarbejdere det drejer sig om. Nedenstående kode viser hvordan dette bestemmes.

```
boolean update = false;
String sql;
for (int i=0;i<defaultEmpID.length;i++) {
    if (defaultEmpID[i] == empID) update = true;
}
```

Hvis variabelen "update" er false, betyder det at den valgte medarbejder ikke er at finde blandt de seneste valgte medarbejdere, og derfor skal der indsættes en ny tuppel i tabellen. Nedenstående kode viser hvordan dette foregår.

```
if (!update) {
    sql = "INSERT INTO RecentEmpSearch values( "+
        "'"+ userID +"' , "+
        "'"+ empID +"' , "+
        "'"+ fullName +"' , "+
        "sysdate )";
}
```



Hvis variabelen "update" er true, betyder det at den valgte medarbejder har været valgt før, og derfor skal tabellen opdateres. Nedenstående kode viser hvordan dette foregår.

```
else {  
    sql = "UPDATE RecentEmpSearch "+  
        "SET timeStamp = sysdate "+  
        "WHERE userID = '"+ userID +"' "+  
        "AND searchedForID = "+ empID;  
}
```

SQL-sætningen udføres ved at benytte DBAccess referencen.

```
dbAccess.setStatement(sql);
```

4.5 Input-argumenter

Alle forespørgelser fra brugeren til MAOV-systemet, går til samme adresse. Hvilken af de forskellige sider der ønskes er specificeret i input-argumenter, på samme måde som CGI/ASP/PHP får argumenter med fra brugeren. Her er et eksempel på en URL, der viser hoved-menuen i MAOV:

- <http://stdemo.dk.oracle.com/ptg/rm?PAoid=308&ACTION=GoMainMenu&SUBACTION=Start>

I eksemplet er der argumenterne:

- PAoid - er styret af PtG, så den kommer vi ikke mere ind på.
- Action - Det grundlæggende action der skal udføres.
- SubAction - Det underliggende action

I Portal-to-Go kan man sætte links op i et hieraki, og på den måde skabe sin egen personlige menu. Det gøres ved at specificere hvilken "Adapter" der skal bruges, og hvilke input-argumenter den skal modtage. På den måde kan man i sin egen personlige menu sætte links op til vilkårlige sider i MAOV-systemet ved at kende de forskellige actions og subactions. Her er derfor en beskrivelse af hvor de forskellige kombinationer af Action og SubAction fører hen.

ACTION	SUBACTION	Beskrivelse:
GoMainMenu	Start	Viser hoved-menuen, med de tre punkter: Kunder, Medarbejder og Pris-liste.
FindCustomer	Start	Viser de kunder der sidst er blevet søgt på i MAOV, og mulighed for fritekstsøgning på resten.
	ShowCustSearchResult	Viser de kunder der er fundet ud fra fritekstsøgningen. Yderligere argumenter krævet: FromIndex, SearchString.
	ShowCustMenu	Viser navn og adresse for den valgte kunde, og menupunkterne der fører til mere detaljerede oplysninger (Licenser, Aktiviteter, TAR, Kontakter). Yderligere argumenter krævet: CustID, CustCDMID.
Licenses	ShowCustomerLicences	Viser licenser som kunden har, med oplysninger i rækkefølgen: antal licenser - produkt-navn version, system-platform, licens-type. Yderligere argumenter krævet: CustID, CustCDMID, FromIndex.



Activities	ShowCustomerActivities	Viser aktiviteter som kunden er tilknyttet, med oplysninger i rækkefølgen: dato for tilknytning - beskrivelse af aktivitet. Yderligere argumenter krævet: CustID, CustCDMID, FromIndex.
TARs	ShowCustomerTAR	Viser aktive TAR's for en kunde, med oplysninger i rækkefølgen: Severity - (TAR-nummer) Hvem der venter på hvem. Yderligere argumenter krævet: CustID, CustCDMID, FromIndex.
	ShowTARDetails	Viser detaljer om en TAR, i form af: Tar-nr, sidst ændret, problembeskrivelse, kontakt-person, kontakt-tlf, hvem der venter på hvem. Yderligere argumenter krævet: TarID.
Contacts	ShowCustomerContacts	Viser hvilke personer Oracle kan kontakte hos kunden. Viser menu med navne man kan trykke på. Yderligere argumenter krævet: CustID, CustCDMID, FromIndex.
	ShowContactData	Viser kontaktoplysninger (navn, stilling, arbejds-tlf, mobil-tlf) om kontakt-personen. Yderligere argumenter krævet: ContID.
FindEmployee	Start	Viser de medarbejdere der sidst er blevet søgt på i MAOV, og mulighed for fritekstsøgning på resten.
	ShowEmpData	Viser navn og telefon-nr til en medarbejder. Man kan desuden vælge "Flere oplysninger" og at sende vedkommende et action-point. Yderligere argumenter krævet: EmpID
	ShowAllEmpData	Viser flere oplysninger om en medarbejder, i form af: Navn, telefon-nr. til arbejde, mobilen og privat og vedkommendes email-adresse. Yderligere argumenter krævet: EmpID
SendActionPoint	Start	Viser email-addressen på modtageren af et action-point, og en menu hvor man kan vælge hvilket action-point man vil sende. Yderligere argumenter krævet: MailTo, EmpID
	ShowEMailsSent	Sender et action-point som e-mail, og viser en simpel side der viser at den er sendt. Yderligere argumenter krævet: MailTo, MailIndex, EmpID
PriceList	Start	Viser kategorierne af produkter i prislisen som menu-punkter.
	ShowProducts	Viser produkterne i en kategori som menu-punkter. Yderligere argumenter krævet: CategoryID
	ShowPrices	Viser priserne for de forskellige licenser for eet produkt. Yderligere argumenter krævet: ProductID

Dette er en komplet liste over de forskellige sider MAOV-systemet kan vise. Man kan linke til dem direkte fra Portal-to-Go service-designeren, men der er flere af dem som kræver yderligere argumenter end Action og SubAction, så de er ikke velegnet til direkte link. Det er dog ikke umuligt at lave eksempelvis et link direkte til sin favorit-kunde, omend det er lidt besværligt at skulle finde de korrekte ID'er fra databasen.

Hvis man forsøger at give MAOV-systemet argumenter som ikke er beskrevet ovenfor, vil den melde ud til brugeren at argumenterne ikke er genkendt.



4.6 Hvordan ændres systemet

Eftersom det oprindelige udvikler-hold på MAOV ikke vil være til rådighed til ændringer af systemet efter det er færdigt, er her en guide der skal hjælpe et eventuelt senere udviklingshold med ændringer.

4.6.1 Generelt om udviklingen.

Systemet er fuldt ud udviklet i Java, og compilet med JDK 1.2.1 fra Sun. Der skulle dog ikke være noget i vejen for at bruge en anden compiler, som eksempelvis JDeveloper. Efter en ændring skal de ændrede klasser naturligvis recompiles.

Systemet er samlet i en package med navnet `oracle.dk.maov`, hvilket sætter krav til at klasserne skal ligge i nogle under-kataloger, der matcher stien på package'n.

Når systemet compiles skal compileren have adgang til de packages, som systemet bruger. Bruger man JDK, skal stien til dem sættes i environment-variablen CLASSPATH. De packages der er brug for er følgende:

- `panama_core.zip` - Portal-to-Go's hjælpeklasser - (*oracle.panama.**)
- `xmldom.jar` - XML-DOM - (*org.w3c.dom.**)
- `classes111.zip` - Driver til Oracle-databasen - (*oracle.jdbc.driver.OracleDriver*)
- `mail.jar` - JavaMail - (*javax.mail.**)
- `activation.jar` - JavaBean Activation Framework (krævet af JavaMail) - (*javax.activation.**)

Det samlede MAOV-system består af 20 klasser, der alle skal pakkes sammen i een JAR-fil eller ZIP-fil. Til det formål har vi brugt `jar.exe` som medfølger til JDK, med følgende parametre:

- `jar cvf0 MAOVAdapter.jar oracle/dk/maov/*.class`

Den samlede pakke skal derefter uploades til Portal-to-Go, som ikke lå på den samme maskine som MAOV blev udviklet på. Til det blev brugt programmet "Portal-to-Go Service Designer", der er en grafisk applikation, og er en del af klient-siden af Portal-to-Go. I det finder man MAOVAdapteren og vælger "import" for at importere den nye version af MAOVAdapteren til systemet.

Portal-to-Go har en cache med de sider den skal vise. Det betyder at efter man har uploadet sin opdaterede MAOVAdapter, kan man få gamle sider, når man tester den. Man skal derfor tømme Portal-to-Go's cache når man har uploadet den ny version. Portal-to-Go har et web-baseret kontrol-interface hvor man kan gøre netop det, ved at trykke på "zap all".

- `http://stdemo.dk.oracle.com:8090/?req=objects`

Vær opmærksom på at din browser også har en cache med gamle sider, og du risikerer at når du trykker "zap all", vises bare en side fra din lokale cache, hvilket betyder at Portal-to-Go's cache ikke bliver tømt alligevel. Du skal derfor trykke på refresh på din browser, for at sikre at du kalder selve "zap all"-siden og Portal-to-Go's cache bliver tømt.



Den nemmeste måde at teste om systemet virker er med det simple web-baserede interface på adressen:

- <http://stdemo.dk.oracle.com/ptg/rm> eller
- <http://stdemo.dk.oracle.com/papz/login.jsp>

4.6.2 Ændrings-scenarier

Her kommer en række generelle beskrivelse af hvordan det oprindelige MAOV-udviklerhold ville have foretaget nogle ændringer vi forestiller os kunne være nødvendige.

Tilføj en ny funktion. Hver funktion har to klasser. En der bygger XML-dokumentet op, og en der henter data fra databasen. Den nye funktion bør få sine egne klasser. Det er nemmest at kopiere et sæt hvis funktionalitet minder om den nye, og så tilretter dem. Der skal desuden ændres i MAOVAdapter-klassen, så den kan kalde den nye funktion. Desuden skal den sættes ind som menu-punkt et passende sted, eller oprettes som et nyt "alias" i service designeren til MAOVAdapteren med de ACTION- og SUBACTION-argumenter der kalder den nye funktion.

Ændre måden data præsenteres. Hvis det er de samme data som bare skal vises lidt anderledes, bør det være nok at ændre i den klasse der genererer siden. De forskellige data hentes fra data-objekterne med get-metoder, og de kan hentes i vilkårlig rækkefølge.

Tilføj yderligere information til en funktion. De nye data skal hentes fra databasen, så den tilhørende data-klasse skal ændres. SQL-forespørgslen ændres, der skal oprettes nye instansvariabler til de nye informationer. Den nye værdi skal trækkes med ud fra ResultSet'et over i Vector'en, og der hvor vectorens værdier flyttes ud i arrays, skal der tilrettes, så de rigtige værdier kommer i de rigtige variabler. Endelig skal der oprettes get-metoder til de nye data. For at få de nye data vist, skal der også ændres i den klasse der genererer siden med oplysningerne.

Vise om en kunde har licenser før brugeren trykker på licenser Når man har fundet en kunde og kan vælge Licenser, Aktiviteter, TAR og Kontakter ville det være rart at kunne se om de var tomme inden man trykker på dem, eksempelvis markeret med en lille stjerne (*). Det er ikke umuligt at lave denne ændring, men det kræver en omstrukturering af systemet. De informationer er nemlig ikke hentet før der er trykket på dem. De skulle så hentes frem allerede når man finder kunden, hvilket ville give længere ventetid, før kundeoplysningerne kan vises første gang. Det objekt der genererer menu-siden (FindCustomer) har heller ikke fat i de fire data-objekter. De skulle så oprettes deri, og kunne videre-gives til de objekter der skal generere de undeliggende sider. Det er ikke holdbart at hente oplysningerne op fra databasen to gange, men det kunne være en umiddelbart nemmere løsning.

Data flyttes til nye databaser. Hvis det er muligt at hente de samme oplysninger frem, vil det kun være de berørte data-klasser der skal ændres. Hele datahentningen som primært forgår i constructorerne, bør ses efter i sømmene. Det formodes at der stadig er adgang til dem fra MAOVs egen database.

Tilføj nye mulige action-points Dette kræver ikke at koden recompiles. Der skal blot tilføjes de nye actionpoints i "actionPoint" tabellen i MAOVs egen database, hvor de står.

Tilføj ny bruger Brugerne styres af Portal-to-Go, som har en brugervenlig brugerflade til opsætning af brugere. Det er yderligere beskrevet i bilag 8.11 side 102.



4.7 Forslag til forbedringer

Følgende ting blev enten nedprioriteret under implementationen eller er ideer der er kommet for sent, og nåede derfor ikke at komme med i det endelige system.

- **Tilmelding til aktiviteter** - Denne funktion var planlagt, men det var ikke muligt at finde oplysninger i databasen om hvilke der ville være relevant at tilmelde en kunde, og det samlede antal af forskellige aktiviteter er meget stort.
- **Smart søge-funktion** - Denne er beskrevet i mockups'ene i analysen, men blev nedprioriteret, da den kun er en alternativ løsning til noget der virker. Der er dog stadig stort potentiale i denne funktion, da den afhjælper et af de store problemer på en WAP-telefon, nemlig den besværlige indtastning af fritekst. Ideen med funktionen er at alle kunder er sorteret efter deres navne, og man kan så snævre sig ind på den ønskede i en form for træ-struktur.
- **Vise antal fundne ved fritekst-søgning** - Når man kun skriver en del af navnet, er det rart et vide hvormange kunder/medarbejdere der opfylder søge-kriterierne, så man kan beslutte om man vil skrive en lidt længere søgestreng eller bare blade de fundne igennem. Hvis der eksempelvis er søgt på "Peter" og der er fundet 100 der hedder Peter, vil man gerne skrive lidt mere af navnet. Antal fundne vises ikke, da vi ikke kunne finde noget sted det skulle så på det lille display, uden at være i vejen. Det burde dog være nemt at tilføje.
- **Korrekt overførsel af æ, ø og å** - På en WAP-emulator var der problemer med at overføre æ, ø og å i en søgestreng. Det blev forsøgt løst, men virkede ikke. Det er dog kun med den WAP-emulator der er problemer, så det er værd at undersøge hvor udbredt problemet er. WAP-emulatoren udskiftede de danske tegn med koder, som MAOV så kan konvertere tilbage, men det viste sig at de koder ikke blev overført ordentligt, så præcist hvor problemet er, er endnu lidt uvidst. Problemet er teoretisk løst, men det virkede ikke i praksis. Læs mere om vores forsøg i afsnit 5.2 "Design-test" side 64.
- **Overblik over kundeoplysninger** - Når man vælger en kunde, kan man vælge licenser, aktiviteter, TAR'er og kontaktpersoner. Nogle af dem kan dog være tomme, og det er derfor lidt irriterende at have klikket sig ind på dem bare for at få at vide at de er tomme. Det ville være smartere hvis man i menuen kunne se hvilke punkter der er tomme. Det kunne være markeret med en lille stjerne, eller at man bare ikke kunne vælge de punkter der er tomme. Se mere om denne feature i afsnit 4.6 "Hvordan ændres systemet" side 59.
- **Oprydning i Recent-tabellerne** - Der vises kun de 10 senest søgte kunder/medarbejdere. Dem der er ældre gemmes dog stadig i databasen til ingen nytte. Der burde blive ryddet op i de tabeller med jævne mellemrum, for at spare plads.
- **Understøttelse af flere sprog** - Alle ord i systemet er på dansk, og de er skrevet direkte ind i koden. Systemet er dog kun tiltænkt til brug i danmark, men det ville alligevel være pænere, designmæssigt, at adskille teksten fra koden. Det kunne gøres ved at lave et sprog-objekt som indeholder en masse Strings med alle ordene i, som resten af koden så henter tekst fra når den skal bruges. Hvis sproget senere skulle ændres, ville det kun være det objekt der skulle ændres. Tænker man videre, kunne sprogobjektet hente alle sine ord fra en database under oprettelse. På den måde behøver man ikke recompile noget, for at ændre sproget, og man kunne have flere sprog i databasen samtidig, som kan vælges ved opstart, individuelt for hver enkelt bruger.
- **Håndtering af SQLExceptions** - Når der hentes data med JDBC, kan der blive kastet nogen Exceptions. Når en sådanne bliver fanget, bliver der ikke gjort noget ved det. Det resulterer i NullPointerExceptions, eftersom de værdier der skal sættes ud fra databasen ikke bliver sat. Den NullPointerException bliver fanget og skrevet ud til slut-brugeren. Det er ikke nogen pæn løsning. I stedet burde systemet fortælle brugeren på en pæn måde at der er sket en fejl, under tilgangen til databasen. Det kan gøres ved at skrive fejl-meddelelsen i de variabler der alligevel skrives ud til



brugeren, eller kalde en generel metode der genererer en pæn og sigende fejlmeddelelse, der sendes til brugeren.

- **Tilbage-menupunkter på samtlige sider** - En WAP-telefon har en back-knap, så man kan altid komme tilbage til hvor man var. Man kan dog ikke garantere at en sådanne back-knap findes på alle enheder. Der burde derfor være menu-punkter der kan lede brugeren tilbage til forrige sider. Det vil også gøre brugen mere intuitivt forståelig, da brugeren så udelukkende skal vælge menu-punkter, og ikke til at bruge alle mulige andre knapper.
- **Hvilke personer der er tilmeldt aktiviteter** - Som det er kan man kun se at et firma er tilmeldt en aktivitet, og ikke hvilke personer det drejer sig om. Et link direkte til firmaets kontaktpersoner kunne give direkte adgang til telefonnumre på dem der er tilmeldt aktiviteter.
- **Opsætning gemmes i DB-tabeller** - Der vises 10 punkter på hver side der er for stor til en WAP-telefon. Hvis tallet 10 skal ændres skal systemet re-compile. Det ville være smartere hvis sådanne opsætningsværdier var gemt i en tabel i databasen. På den måde kunne hver enkelt bruger også have sine individuelle indstillinger.
- **DB-links direkte til tabeller** - Når der hentes data fra databasen sker det med DB-links. De links er dog til hele databaser, og ikke til tabeller. Hvis en tabel blev flyttet til en anden database, skal koden ændres. Det ville være smartere hvis der var oprettet synonymymer i MAOV's egen database til de forskellige DB-links. Hvis tabellerne en dag blev flyttet, ville det være nok at ændre i de synonymymer.
- **Undgå kode-gentagelse** - Der er flere steder i koden, hvor store mængder kode er gentaget med kun få ændringer. Det gør sig eksempelvis gældende i alle entitetsobjekterne der flytter data fra ResultSet til Vector og videre til array's. Det ville være smart at lave en generel hjælpemetode, der kan bruges alle stederne. Det gør sig også gældende for den kode der deler siderne op, så der kun vises 10 linier ad gangen. Det er besværligt at vedligeholde et system med meget kopieret kode.
- **Søg kontaktperson ud fra navn, ikke firma** - Man kan kun søge kontaktpersoner ud fra hvilket firma de er tilknyttet, i den overbevisning at man ikke er i tvivl om hvilket firma en person kommer fra, som man vil i kontakt med. Det kan dog i visse tilfælde være en fordel at søge mellem samtlige kontaktpersoner, uden først at skulle vælge firmaet.
- **CC-email til sig selv, når action-points sendes** - Dette var faktisk et ønske fra brugerne, men blev ikke implementeret. Der sendes kun en e-mail til modtageren af action-pointet, og ikke til afsenderen.

Ud over disse forslag til forbedringer, er der forslag til eksterne funktioner i bilag 8.12 side 103.

4.8 Konklusion

Systemet er blevet implementeret næsten fuldt ud. Hvad der mangler vil vise sig i de tests der er beskrevet i efterfølgende kapitel. Vi blev færdige en lille uge før tid, og systemet virker tilfredsstillende.

Forskellige interessante aspekter af koden er blevet dokumenteret, og kan forhåbentlig hjælpe et fremtidigt udviklerteam med at forstå systemets opbygning. Selve program-koden er kommenteret og dokumenteret i henhold til JavaDoc-standarden. Source-kode og JavaDoc er inkluderet på CD'en der er vedlagt rapporten.

Det implementerede system kræver Portal-to-Go for at kunne bruges. Der er dog nogle test af systemet i main-metoderne i de forskellige klasser. De er dog kommenteret ud, i den endelige version.

Eftersom systemet ikke kan afprøves i funktion, udenfor Oracle's servere med PtG og de nødvendige databaser, kan et udsnit af det endelige system ses iform af screenshots i bilag 8.10 side 99.



5. Test

5.0.1 Indledning

I dette kapitel beskrives de forskellige former for test der er udført på MAOV-systemet for at teste om det virker efter hensigten. Der er en kort forklaring af hver enkelt type test, så det kan læses af enhver.

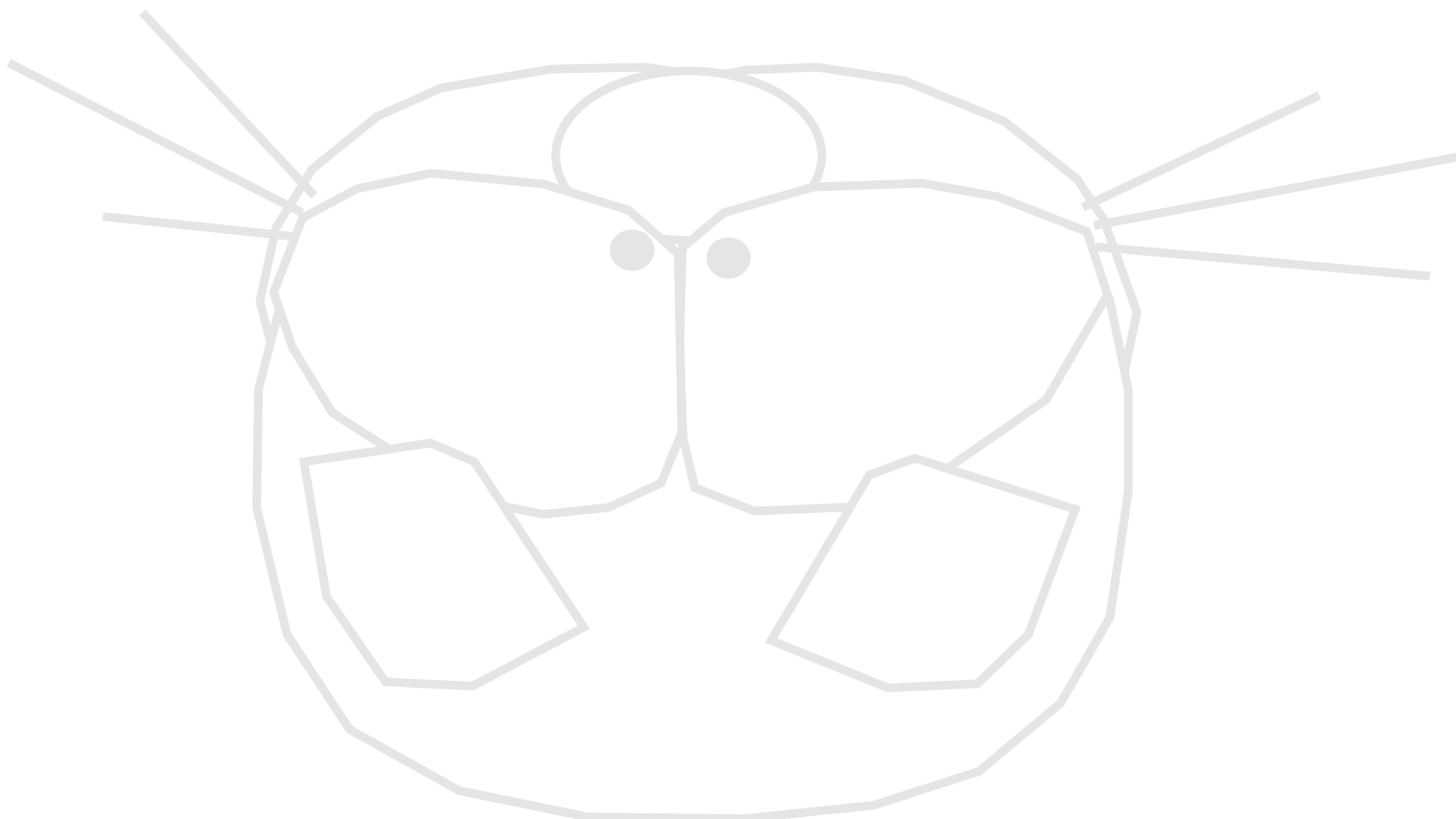
Usecase-test er den mest interessante test, da den tester systemet op imod en tidlig specifikation af systemet. De andre tests her er snarere en beskrivelse af hvordan de er foregået.

5.0.2 Test-plan

Systemet er testet på forskellige niveauer, som man kender det fra "V-modellen" for test. De forskellige test udføres med de forskellige dokumenter i rapporten som grundlag. Vi har planlagt følgende tests:

- **Teknologi-test** - Udført tidligt i projekt-forløbet, for at teste om den nødvendige teknologi er til rådighed..
- **Design-test** - Udført før implementering, for at test nogle design-aspekter.
- **Whitebox-test** - Udført under implementering, når der opstod problemer.
- **Blackbox-test** - Udført iform af main() i alle entitets-objekter.
- **UseCase-test** - Udført ved at følge usecases med systemet.
- **Bruger-test (accept-test)** - Udført ved at lade brugere afprøve systemet.

Ikke alle test er klaret igennem 100% fejlfri, men de fundne fejl er ikke kritiske, og der har ikke været ressourcer til at rette dem. De er samlet i afsnit 4.7 "Forslag til forbedringer" side 61.





5.1 Teknologi-test

Allerede meget tidligt i projektet begyndte vi at programmere et system som bruger de samme systemer (PtG og DB'en) som det endelige produkt, for at teste om det virkede, og for at lære at bruge teknologien. Det resulterede i MAOVAdapterProto (prototype) der kunne følgende ting:

- Vise sider gennem Portal-to-Go, genereret som XML i java, med PtG's hjælpeklasser.
- Linke mellem forskellige sider, med argumentet ACTION.
- Hente data fra Oracle's database, og vise det gennem PtG
- Svare på en søgeform gennem PtG, og viser data fra DB'en
- Modtage forskellige argumenter

MAOVAdapterProto bygger på noget kode af Asger Jensen, så vi startede ikke på bar bund, og vidste faktisk at teknologien var tilstrækkelig. Det viste sig at være relativt nemt at udvikle videre på, og testen viste hvordan systemet skal kommunikere med Portal-to-Go, hvilket har lagt grundlaget for det videre design.

5.2 Design-test

For at opnå et holdbart design, har design-mæssige spørgsmål været testet i små test-klasser. De er ikke brugt direkte i MAOVAdapteren, men har påvirket hvordan systemet er implementeret.

- **DBTest** - Tester om vi fra Java kan tilgå den database vi har fået tildelt. Det var ikke noget problem.
- **TestCharacterConverter** - Når systemet blev testet i en WAP-emulator fungerede det ikke, hvis der blev indtastet æ, ø eller å i et søge-felt. Det viste sig at de blev skiftet ud med en række andre tegn (eks. æ = %E6). Denne klasser tester om vi på en enkel måde kan konvertere tilbage til de rigtige bogstaver. Det virkede her i test-klassen, men implementeret i systemet virkede det ikke. De erstattede koder blev slet ikke sendt ordentligt med, fra WAP-emulatoren, så det blev droppet at konvertere koderne tilbage til de danske tegn.
- **TestGetField** - Dette er egentligt en hjælp til en anden projekt-gruppe, der skulle kunne oprette et Color-objekt ud fra navnet på farven fra en String. Det lagde grunden for ideen om test-systemet JavalnstantTester, som er udviklet lidt sideløbende med Project MAOV.
- **TestPAPrimitive** - Vi havde to forskellige beskrivelser af hvilke XML-tags Portal-to-Go accepterer, men en mistanke om at Java hjælpeklassen (PAPrimitive) kun understøttede de gamle og mere begrænsede tags. Den understøttede kun de gamle. Denne test bygger iøvrigt også på erfaringerne med TestGetField.
- **TestPassingObjects** - get-metoder laves for at beskytte variabler, men hvis man ikke laver get-metoderne ordentligt, beskytter de ikke alligevel. Denne klasse tester hvad der er beskyttet med get-metoder. Svaret var primitiver, og objekter der ikke kan ændres såsom String.
- **TestPtGXML** - Tester hvordan den XML ser ud som Portal-to-Go modtager. Den er lavet for at få et eksempel der kunne bruges i beskrivelse af PtG's XML, der endte som bilag 8.9 side 97.
- **TestTimeTakenCustomerData** - Tester tiden det tager at hente data om en kunde fra databasen. Det var som svar på, om samtlige oplysninger om en kunde, inklusiv de fire underpunkter, skal hentes op af databasen straks når brugeren vælger en kunde, eller de fire underpunkter først skal hentes, når man trykker på dem. Resultatet var at det tog for lang tid at hente alle data på een gang. Specielt oplysningerne om TAR var tidskrævende.



- **TestTryCatchFinally** - I den prototype Asger Jensen havde lavet, brugte han "finally", som vi ikke vidste hvad gjorde. Derfor testede vi det i denne klasse.
- **TimeTaker** - En gammel genbrugelig testklasse der bruges i TestTimeTakenCustomerData, til at måle tidsforbruget for noget kode.
- **SQLMachine** - Vi havde problemer med at installere Oracles SQL+ på en lille computer, så vi udviklede dette program som erstatning. Vi har haft meget gavn af det, da det har en bedre brugerflade end SQL+.
- **JavalnstantTester** - Dette er lidt af et sidestykke, da det viste sig at blive for kompliceret at udvikle, og nåede derfor ikke at blive færdigt til at kunne bruges ordentligt. Det bygger på erfaringerne fra TestGetField. Dets formål er at man kan kalde metoder i objekter i vilkårlig rækkefølge mens programmet kører, for at teste dem, uden at skulle omskrive en test-main-metode flere gange.

5.3 White-box test

En komplet whitebox-test tester samtlige muligheder i selection og iteration i programmet. Vi har ikke udført en sådanne komplet test, da de ressourcer det ville kræve ikke modsvarer vigtigheden af at systemet er 100% stabilt. Til gengæld er systemet forsøgt testet mod forskellige situationer på en usystematisk måde under implementationen, for at fange eventuelle fejl. Når der er opdaget fejl, er de blevet lokaliseret med at udskrive test-værdier forskellige steder i koden, og på den måde se om de forskellige iterationer og selektioner nås. Da der kun er whitebox-testet usystematisk, kan vi ikke garantere 100% stabilitet, men systemet er sikret imod de fleste typer fejl, såsom at der kommer null-værdier fra databasen.

5.4 Black-box test

En blackbox-test tester et objekt udefra ved at kalde de forskellige metoder med alle mulige argumenter. MAOV-systemet er ikke et generelt system hvor de enkelte klasser skal kunne bruges af andre uden tilpasning. Det er derfor ikke nødvendigt at teste for samtlige destruktive måder at anvende klasserne på. Det skal derimod testes om de enkelte klasser kan det som de skal kunne i forbindelse med MAOV-systemet.

Blackbox-testen udføres op imod design-modellen, for at sikre at det implementerede svarer til det designede. Eftersom MAOV-systemet er designet og implementeret parallelt med round-trip, kan de to dog ikke adskille sig meget fra hinanden. Der er derfor ikke nogen direkte sammenligning, som ved usecase-testen senere.

MAOV-systemet er opbygget i to lag, der testes på hver sin måde. Det underliggende lag testes før det overliggende, da det er for tidskrævende at lave et "fake"-lag til test af det overliggende.

Det nederste lag af de to, henter data fra databasen. Blackbox-testen består i test-main-metoder i hver klasse der henter test-data fra databasen og skriver dem ud til konsollen (MS-DOS-vinduet). Hver enkelt klasse der henter data fra databasen er testet på denne måde.

Det overliggende lag som genererer XML-dokumenter ud fra data der er hentet fra databasen, er primært testet ved at køre systemet gennem PtG, og hente de relevante sider frem. Hvis der har været problemer, har vi testet problem-siden med main i MAOVAdapter-klassen der kan udskrive XML'en for en side angivet med ACTION og SUBACTION parametrene.

Blackbox-testen er ligesom whitebox-testen udført løbende under implementeringen. Det betyder at der ikke er nogen testresultater, da de fejl der er fundet er blevet rettet umiddelbart efter.



5.5 UseCase-test

Denne test skal afsløre om systemet kan det som det rent faktisk skal kunne. Man følger usecase-beskrivelserne og bruger systemet som det står beskrevet.

5.5.1 Søg kundeoplysninger ()

Det er muligt at søge en kunde frem, for at se oplysninger om denne. I use-casen står beskrevet at systemet ved hvilke kunder en bruger har, og foreslår dem først. Det ved det ikke direkte. Det ved dog hvilke 10 kunder der er søgt frem sidst, af den bruger, og det er dem der listes op som direkte links, udover at der er fritekst-søgning. Selvom systemet ikke virker helt som use-casen, ændrer det ikke på anvendeligheden.

5.5.2 Se licenser

Det er muligt at se samtlige, ikke terminerede licenser for en kunde. Samtlige ønskede oplysninger om licenser vises.

5.5.3 Se oplysninger om aktuelle sager

Denne use-case kan udføres fuldt ud. Alle de ønskede oplysninger hentes frem. Funktionen er dog noget langsom, da det er en kompliceret SQL der henter data ud.

5.5.4 Se marketingsoplysninger - evt. tilmeld -

Denne use-case kan følges, men med modifikationer, og tilmelding mangler fuldstændig. Kilde-dataene skelner ikke mellem mailinglister og seminarer, men ser det hele som aktiviteter, som en kunde er tilknyttet. MAOV-systemet kan derfor heller ikke skelne. Der vises en kort beskrivelse af hver aktivitet, som er aktiv, og de er sorteret efter datoen, som de er tildelt på. Det var ikke muligt at finde fremtidige seminarer, så der er ikke noget at tilmelde. Selvom usecasen ikke følges helt, kan funktionen dog stadig bruges.

5.5.5 Find kontaktperson hos kunde

Denne use-case kan følges fuldt ud. Hvis et firma har flere adresser, er det dog indsat i systemet som forskellige firmaer med samme navn, så det kan godt være lidt besværligt at finde den rette afdeling. Dog vises bynavnet for de forskellige firmaer hvor muligt. Ellers er alle de ønskede oplysninger med.

5.5.6 Søg medarbejderoplysninger

Denne use-case kan udføres fuldt ud. Man kan søge blandt samtlige af Oracle DK's medarbejdere.

5.5.7 Find tlf-nr. til kollega

Denne use-case hænger tæt sammen med "Søg medarbejderoplysninger", og virker også helt efter hensigten. Der vises de telefon-numre der er i databasen, som er henholdsvis arbejds-telefon, mobil-telefon og privat-telefon. Desuden vises eMail-adressen.



5.5.8 Tildel action-point til kollega ()

Man skulle kunne tildele action-points, fra forskellige steder i systemet, og kunne vælge imellem forskellige action-points. Vi kunne dog ikke finde andre action-points end "ring", og det er kun tilknyttet i systemet, når man har fundet en medarbejder. Ellers virker det fint.

5.5.9 Check e-mail

Denne use-case blev ikke en direkte del af MAOV-systemet, idet den allerede var lavet af Asger Jensen. Den virker dog, og kan integreres med MAOV.

5.5.10 Find pris ()

Denne use-case gør det den skal, men man skal være opmærksom på at pris-oplysningerne ikke er garanteret opdaterede i den første periode. Det vil de dog blive, når Oracle DK har udviklet et lille program til at opdatere dem med, og en sekretær e.lign bliver sat til at opdatere dem (hvilket vil ske meget snart). Der er de oplysninger der skal være, og produkterne er delt op i grupper.

5.5.11 Test-resultat

Systemet har bestået usecase-testen, selvom der er små detaljer der adskiller det system der er beskrevet i usecase'ene og det der er implementeret. De små detaljer gør på ingen måde systemet uanvendeligt til det det var tiltænkt. Der var desværre ikke tid til at rette de små fejl.

5.6 Bruger-test

Denne test skal afsløre om brugerne er tilfredse med systemet. Det gøres ved at lade de endelige brugere af systemet forsøge at bruge det.

Projektet har her hos Oracle DK, status som uofficielt og uden support. Det har derfor ikke været muligt at lave en omfattende brugertest. I stedet er systemet blevet afprøvet af nogle af de personer vi har haft kontakt med under projekt-forløbet. Deres udsagn har desuden været grundlag for om systemet er accepteret (accept-test). Her er hvad de forskellige personer har givet af respons på bruger-testen.

5.6.1 Christian Fabricius

Christian var den første til at afprøve systemet (se mødereferat af 17.10.2000 side 110). Dengang var systemet endnu ikke som i sin endelige form, idet meget af det Christian pointerede som u hensigtsmæssigt nåede at blive rettet. Her er de punkter, og deres status.

- Søgning på kunder for langsom - Den store database er langsom i sig selv, men læsningerne er blevet begrænset.
- Når kunder listes, vis hvilke der har licenser - Denne oplysning er ikke fremskaffet på det tidspunkt, så man er nød til at vælge licens-oplysninger for kunden. Ikke rettet.
- Skriv bynavn med når kunder listes så man kan skeldne afdelinger - Er rettet.
- Samme aktivitet vises nogen gange flere gange - Skyldtes at flere personer fra samme firma kunne være tilmeldt den samme aktivitet. Er blevet rettet.
- TARs skal præsenteres anderledes - Er rettet.
- Prisliste mangler - Er tilføjet.
- Eneste relevante action-point: Ring til mig! - Er tilføjet.



Vi har snakket med Christian efter rettelserne, og han er tilfreds med systemet. Det er accepteret som brugbart.

5.6.2 Søren Hebsgaard

Søren prøvede systemet som det endeligt kom til at se ud, og havde ikke noget at indvende. De data som MAOV-systemet henter og viser er noget rodede, og derfor kan det være lidt besværligt at finde de ønskede data. Søren mente dog ikke at det var noget MAOV-projektet kunne gøre mere ved. Det er et spørgsmål om at Oracle rydder op i deres databaser. Søren er derfor tilfreds med MAOV-systemet, og har accepteret det som det er.

5.6.3 Asger Jensen

Asger fandt også systemet brugbart. Han bemærkede specielt at der er en del funktioner som ikke var implementeret i den prototype han var med til at lave før MAOV-projektets start. Der hentes data fra flere forskellige databaser, og at systemet husker de sidst søgte kunder/medarbejdere er en feature han ser som vigtigt for brugbarheden på en WAP-telefon. Han pointerede dog følgende:

- Der mangler menupunkter der fører brugeren tilbage til tidligere sider. Dette er undladt fordi WAP-telefonen som standard har en back-knap, som virker fint i MAOV-systemet. Asger mente dog at brugeren kan blive forvirret hvis ikke det også er som menu-punkter på selve siden.
- Man kan ikke se hvilken person der er tilmeldt en aktivitet. Det kunne være rart at vide.
- Der burde være en forbindelse imellem kontakt-personer og aktiviteter, så man nemt kunne se hvem der var tilmeldt hvad, og den anden vej fra, hvad de enkelte personer er tilmeldt.

På trods af disse konstruktive forslag til forbedringer, er systemet accepteret.

5.7 Konklusion

Systemet har klaret sig igennem samtlige tests med tilfredsstillende resultat. En del af testene er foretaget under implementeringen, og fejl er derfor umiddelbart efter rettet.

Samtlige planlagte funktioner på nær een er implementeret og virker efter hensigten. Den ikke implementerede funktion er tilmeldelse af kunder til seminarer. Det var desværre ikke muligt at finde i databasen hvilke seminarer, der var afholdt, og hvilke der var fremtidige.

De få brugere der testede systemet accepterede det. Det er ikke et kritisk system som de er afhængige af, men snarere et supplement til deres muligheder. Det gør det svært at få folk til *ikke* at acceptere det. Der var desværre ikke ressourcer til at teste systemet gennem en længere tids brug, så vi må nøjes med denne simple accept-test af brugerne.

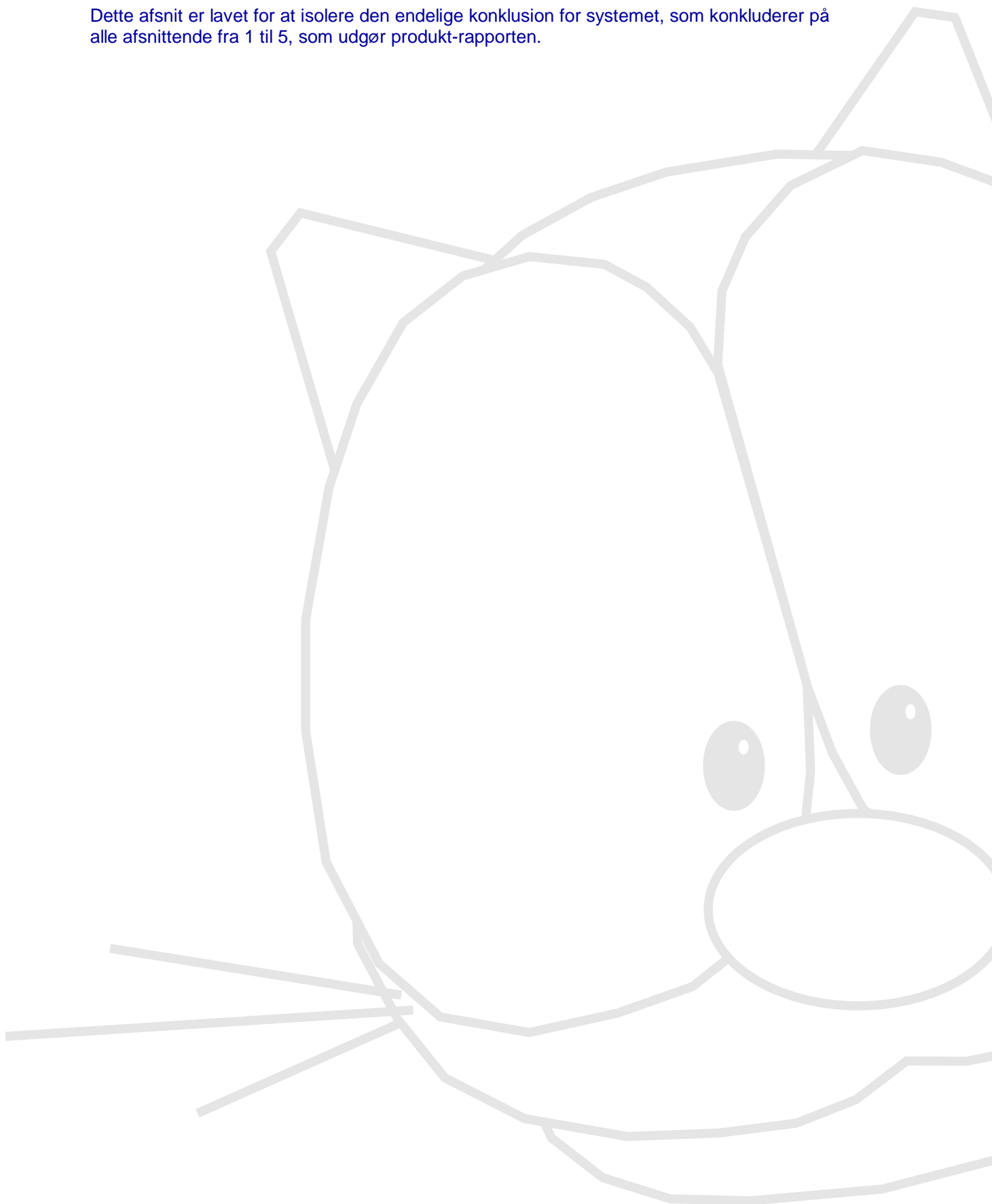
I mock-ups'ene af systemet er beskrevet en alternativ måde at søge på end fritekstsøgning til kunder og medarbejdere. Denne ekstra-feature blev slet ikke implementeret, og det blev ikke opdaget i de tests af systemet der er udført. Det skyldes at det kun er en ekstra-feature, der egentligt ikke er nødvendig for at systemet skal virke.

Den endelige konklusion er derfor: Samtlige tests - OK



6. Afslutning

Dette afsnit er lavet for at isolere den endelige konklusion for systemet, som konkluderer på alle afsnittene fra 1 til 5, som udgør produkt-rapporten.





6.1 Konklusion på systemet

Projektet er blevet fuldført. Opgaven er løst, og det endelige resultat er et dokumenteret og fungerende system, som medarbejderne hos Oracle Danmark kan drage nytte af.

Omfanget af systemet er blevet afgrænset af flere omgange, af forskellige tekniske og organisatoriske årsager. Det betyder der er grundlag for at videreudvikle og forbedre på systemet i et senere udviklingsforløb. Rapporten dokumenterer de forskellige aspekter vedrørende design og implementering, så det burde være relativt nemt for et nyt udviklerteam at overtage udviklingen. Endvidere er der en række forslag til forbedringer af systemet, som vi allerede nu har fundet relevante, men som ikke blev implementeret i denne version.

MAOV-systemet kan bruges til at få "Mobil Adgang til Online Virksomhedsoplysninger", som det som udgangspunkt er udviklet til. Derudover kan det bruges til at demonstrere nogle af de muligheder som Oracle's Portal-to-Go tilbyder, i forbindelse med salg af denne til Oracle's kunder.

Produktet vil blive sat i drift hos Oracle Danmark, men kun som en uofficielt service, uden support. Det vil dog ikke umiddelbart have den store betydning for de endelige brugere der bare skal bruge det. Man kan dog risikere at de systemer som MAOV-systemet tilgår bliver ændret, og den manglende support af MAOV-systemet vil betyde at det ikke vil blive tilpasset de eventuelle ændringer, og derfor kan blive ubrugeligt. Man kan dog håbe at hvis systemet får succes, kan det bruges som erfaringsgrundlag til et officielt system, som giver hele Oracle "Mobil Adgang til Online Virksomhedsoplysninger".



7. Proces

7.0.1 Indledning

Dette kapitel beskriver processen under udarbejdelsen af projektet. Det kan læses hvis man er interesseret i hvilke vilkår der blev arbejdet under, og hvordan projektet er forløbet.

Der er beskrevet opgavens karakter med mål og formål, de kritiske succesfaktorer for projektet, en tidsplan for projektet og en gennemgang af hvordan den er fulgt. Der er beskrevet kort hvilke værktøjer fra undervisningen der er anvendt, og hvad der er gjort for at sikre kvaliteten af produktet og dokumentationen. Endelig er der en meget kortfattet dagbog, hvor man kan følge hovedpunkterne under projektets forløb dag for dag.



7.1 Opgavens karakter

Vi har i samarbejde med Oracle udarbejdet en opgave-formulering før projektets officielle start, så projektets karakter hælder imod et konstruktions-projekt. Kravene skulle dog uddybes væsentligt i en kravmodel i samarbejde med brugerne. Vi har desuden skullet benytte, for os, ny og ukendt teknologi, så det er ikke bare simpel konstruktion.

Projektet er et skoleprojekt og udføres derfor for at tilfredstille de studerendes behov for at få praktisk erfaring som eksamensgrundlag. Derudover laves projektet for en virksomhed som forventer et resultat der også kan benyttes efter projektets afslutning. De studerende skal altså bruge udviklingsprocessen mens virksomheden skal bruge resultatet. Det giver to sæt formål og mål, som dog ikke vil være modstridende, men som begge skal opfyldes. Der arbejdes dog primært på at opfylde firmaets behov, og de studerendes behov vil følgelig blive opfyldt, da projektet er defineret fra start af for at opfylde dem.

7.1.1 Formål og mål for virksomheden

- **Formålet** for projektet er at give Oracle Danmarks medarbejdere en lettere hverdag ved at give dem adgang til nogle services via en mobil enhed - i første omgang en WAP-telefon.
- **Målene** der skal opfyldes for at projektet er udført:
 - Undersøge hvilke behov Oracle Danmarks medarbejdere har for et mobilt informationssystem for samtlige afdelinger.
 - Give adgang til de ønskede services der egner sig til WAP.
 - Få løsningen dokumenteret, så fremtidig vedligeholdelse kan udføres

7.1.2 Formål og mål for de studerende

- **Formålet** for projektet er at udføre et komplet og veldokumenteret udviklingsforløb, som grundlag for hovedopgave-eksamen for datamatikerstudiet.
- **Målene** der skal opfyldes for at projektet er udført:
 - Brugernes behov skal analyseres.
 - En løsning skal designes og implementeres.
 - Samtlige af aspekterne under projektet skal dokumenteres.

Ser man på målene for de to grupper vil man se at de minder meget om hinanden, hvilket gør at vi ud fra at gå efter et sæt mål, kan opfylde begge formål. De lægger dog fokus forskelligt, så det er ikke muligt at se bort fra hverken det ene eller det andet sæt.

7.1.3 Ressourcer

Projektet har haft følgende ressourcer til rådighed:

Menneskelige ressourcer:

- Egen arbejdskraft: 12 uger fuld tid, to mand.
- Oracles medarbejdere: korte møder efter behov (inkl. teknisk kompetence).
- Vejledere på skolen: korte møder efter behov.

Materielle ressourcer:

- Egne computer til udvikling af klient-delen af systemet og dokumentation.
- Oracles server, til at køre den tunge del af systemet på.
- Sted at arbejde hos Oracle.
- Den nødvendige Oracle-software der måtte være brug for.



7.1.4 Skærsilden

I starten af projektet blev vi udsat for et intro-forløb der gik ud på at gennemføre "Skærsilden".

Skærsilden er navnet på et mini-projekt som nye medarbejdere hos Oracle skal gennemføre. Formålet er at de nye bliver kendt i virksomheden, og består af en masse opgaver som skal udføres i huset. Det kan være at komme i snak med direktøren, printe ud på forskellige printere, lokke penge ud af kassen og eller bare snakke med folk fra alle de forskellige afdelinger i Oracle Danmark.

Da vi ikke er fast-ansatte valgte vi at lave en "lite"-version hvor vi valgte de opgaver ud som vi mente var relevante. Det har stadig haft den positive effekt at vi nu har fået et overblik over Oracle Danmark, på en sjov måde. Skærsild-rapporten er ikke medtaget i den endelige rapport, da det er en 11 siders intern rapport, der ikke har relevans for selve projektet. Den er dog vedlagt på CD'en.

7.2 Kritiske succesfaktorer

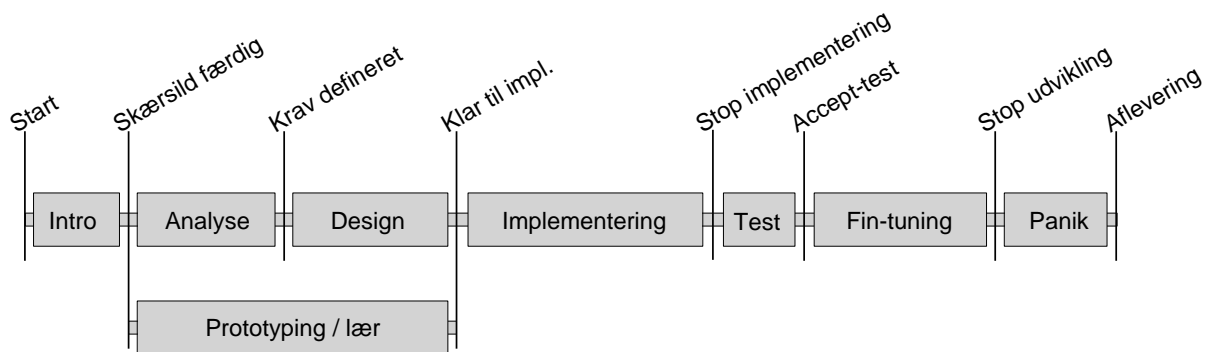
Her er beskrevet de faktorer der har haft betydning for projektets gennemførelse, hvordan de er blevet opfyldt, og hvad vi ville have gjort hvis de ikke var blevet opfyldt.

- **Arbejdspladser** - For at kunne udvikle et system for Oracle er det en meget stor fordel at få plads til at arbejde ved, hos Oracle. Det vil gøre det væsentligt nemmere at komme i kontakt med de forskellige mennesker i virksomheden. Vi fik fra starten af lov til at arbejde i et stort rum, hvor folk kommer og går jævntligt. Vi blev dog flyttet til et mindre lokale, hvor fire forskellige grupper af studerende skulle arbejde på deres afgangsprojekter. Der har altså ikke været de store problemer med at få tildelt arbejdspladser. Alternativet ville være at arbejde på egen bopæl, hvilket ville gøre det væsentligt mere ressourcekrævende at komme i kontakt med medarbejdere hos Oracle.
- **Skaffe nødvendigt udviklingsudstyr** - Oracle var ikke villige til at stille hardware til rådighed til udvikling af MAOV-systemet. Vi var derfor nødsaget til at medbringe egne computere. Det var dog et problem idet, at hvis vi selv skulle hoste en server til database og Portal-to-Go skulle vi investere i større maskiner. Det lykkedes dog at få lov til at bruge en server der normalt bruges til test af diverse ting. Vi manglede dog stadig nogle klient-maskiner til at udvikle på, op imod serveren, og der blev vi nød til at medbringe egne maskiner. I den første lange tid, nøjedes vi med en enkelt gammel bærbar maskine, men senere måtte vi medbringe vores store maskiner. Dette er i sig selv noget af en nødløsning, men et sidste alternativ ville være at flytte udviklingen ud til uddannelsesstedets computere.
- **Skab forbindelse mellem Portal-to-Go og WAP-telefon** - Det var afgørende hurtigt at vide at den nødvendige teknologi var til rådighed, og at vi forstod at bruge den. Derfor lavede vi tidligt i projekt-forløbet en test-applikation der skulle bevise at systemet kunne laves. Det lykkedes uden de store problemer. Alternativet ville være at finde en alternativ teknologisk platform at udvikle systemet på, eventuelt direkte i WML, med Servlets.
- **Skaffe behov fra brugerne** - Udført i form af interviews med udvalgte brugere. Det var lidt besværligt at finde personer der ville deltage, og vi blev derfor nød til at definere kravene ud fra udsagn af en relativt lille gruppe mennesker. Der blev interviewet 6 personer ud af de 15 som vi havde planlagt at interviewe. Hvis det ikke havde været muligt at skaffe behovene fra brugerne, ville vi være nødsaget til bruge andre foranalyse-metoder. De fleste bygger dog på brugerinterviews så en sidste løsning måtte være selv at finde på nogen.



- **Kan vi opnå tilstrækkelig sikkerhed** - Eftersom systemet kan bruges til at finde/visе fortrolige oplysninger, er det vigtigt at kunne opnå tilstrækkelig sikkerhed, så systemet ikke kan misbruges af uvedkommende. Det viste sig dog at Oracle har tilstrækkelig sikkerhed i forbindelsen til serveren som systemet kører på. Det var derfor ikke nødvendigt for MAOV at bekymre sig yderligere om sikkerheden. Hvis ikke der var den nødvendige sikkerhed, kunne vi alligevel have udviklet systemet, men det ville aldrig kunne sættes i anvendelse.
- **Kan vi få adgang til de nødvendige data** - Systemets primære formål er at vise oplysninger fra centrale databaser på en mobil enhed, så det er meget kritisk om vi kan få adgang til de data. Der blev foreslået en del funktioner til systemet, hvoraf omkring en tredjedel måtte opgives, fordi der ikke kunne skaffes adgang til de relevante data. Der var dog rigeligt til at opbygge et fornuftigt system. Alternativet ville være at bygge systemet op på fiktive data.

7.3 Ekstern projektplan



Startsituation

Opgavebeskrivelse er udarbejdet.

7.3.1 Fase: Intro / skærsild

For at sluse folk ind, er der et intro-forløb.

Der udarbejdes:

Skærsild-rapport.

Faselinje 18.08.2000: Skærsild færdig

Udarbejdet skærsild-rapport der beskriver diverse ting om Oracle Danmark.

7.3.2 Fase: Analyse

De specifikke krav skal undersøges hos de forskellige brugere, så der ikke er tvivl om hvad systemet skal kunne.

Der udarbejdes:

Kravmodel med Usecases, PDOM og mockups. Analysemodel med ICE.

Faselinje 13.09.2000: Krav defineret

Alle krav til systemet er beskrevet og godkendt af Oracle.



7.3.3 Fase: Design

Design af systemet, så det er klart til at implementere.
Der udarbejdes:
Klassediagram og sekvensdiagrammer.

Faselinje 22.09.2000: Klar til implementering
Systemet er designet, og designet er detaljeret beskrevet.

7.3.4 Fase: Implementering

Systemet programmeres med diverse tests løbende.
Der udarbejdes:
MAOV-systemet programmeres og koden dokumenteres.

Faselinje 18.10.2000: Stop implementering
Systemet er klar til demonstration.

7.3.5 Fase: Test

Brugerne bliver præsenteret for systemet, og eventuelle fejl bliver noteret.
Der udarbejdes:
Test-resultater.

Faselinje 20.10.2000: Accept
Systemet kan accepteres med/uden modifikationer, eller kasseres.

7.3.6 Fase: Fin-tuning

Eventuelle modifikationer rettes, eller hvis kasseret, så beskrives begrundelsen.

Faselinje 27.10.2000: Udviklings-stop
Ikke flere ændringer til systemet.

7.3.7 Fase: Panik

Der lægges layout på rapporten, og manglende afsnit udarbejdes.

Faselinje 09.11.2000: Aflevering
Definitivt stop på projektet.

7.3.8 Fase: Prototyping / lær

Parallelt med det primære forløb køres en sideaktivitet, med det formål at tilegne os teknisk viden om de ukendte områder. Denne fase kører samtidigt med analyse og designfaserne, og vil bestå af kurser og egne eksperimenter (prototyping).

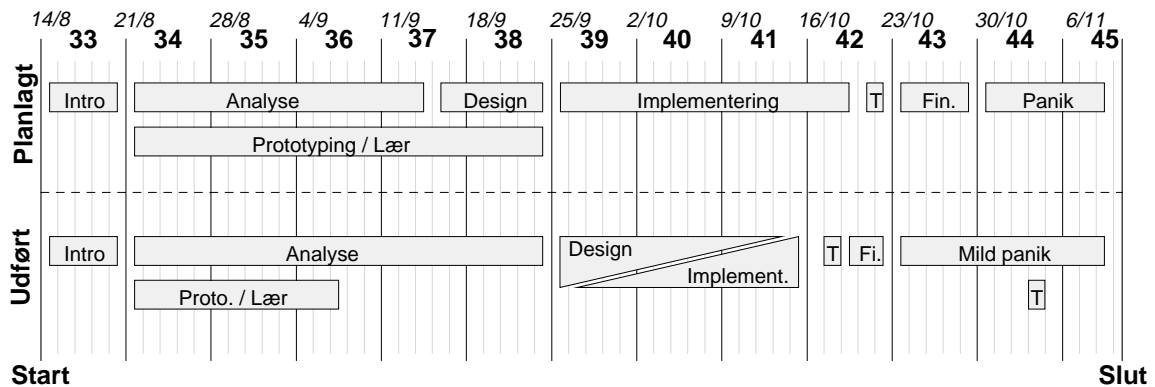
Der udarbejdes:

En fungerende prototype der gør brug af de relevante teknologier der er krævet af projektet.



7.4 Gennemgang af projektplan

Projektet er udarbejdet ud fra den oprindelige projektplan, men de forskellige faser er blevet forskubbet lidt. Der var dog en primær faserække vi ikke ville overskride. Det var skiftet fra at analysere systemet til at implementere det (22/9). På det tidspunkt var designet ikke færdigt som planlagt, så planen blev tilrettet så design og implementering kørte parrallelt i round-trip. Det viste sig at være meget effektivt, og vi blev derfor færdige med implementeringen før tid.



Illustrationen viser tidsforbruget for de forskellige faser, både for den planlagte projektplan, og som projektet faktisk blev udført. I toppen står ugenumrene, og over det står datoerne for hver mandag i ugen. Alle datoer er fra år 2000. Faserne hedder det samme som i den oprindelige projektplan, men kan være forkortede af tekniske årsager.

7.4.1 Hurtig prototyping

Prototyping / Lær-fasen var ment som en parrallel fase med hele analyse- og design-fasen. Der var dog nogle problemer med at få kontakt med folk i starten, så vi brugte mere tid på at studere teknologien end at analysere systemet. Vi blev derfor tidligt færdige med en fungerende prototype der brugte den nødvendige teknologi.

7.4.2 Design flyttet

Vi havde et møde/review med vores vejleder på skolen sidste dag på faserne før implementering. D.v.s at på det tidspunkt lavede vi status på hvor langt vi var. Designmodellen var langt fra færdig, men vi havde lavet en masse overvejelser og kendte teknologien. Vi var derfor ikke langt fra at kunne starte implementeringen. Vores skolevejleder (JKA) fortalte at det idag var anerkendt at designe og implementere sideløbende (round-trip), og vi tilpassede derfor planen til det, med stor success. Læs mere om round-trip i bilag 8.5 side 91.

7.4.3 Hurtig implementering

Vi var færdige med implementeringen en lille uge før planlagt, og fik det testet af en bruger. Fin-tuningen blev derfor også rykket til tidligere. Efter det valgte vi at lægge låg på implementeringen og koncentrere os om dokumentationen af systemet. Der er en sidste test meget sent i forløbet. Det skyldes at vores primære kontaktperson Asger Jensen kom tilbage fra orlov, og vi mente at det var relevant at han også testede systemet. Det var primært ham der havde udviklet deres egen tidlige prototype.



7.4.4 Mild panik

Navnet på den sidste fase er en påmindelse om ikke at gå i panik, fordi der mangler en masse i sidste ende. At denne fase er påbegyndt tidligere gør i sig selv panikken mildere. Der har ikke været panik, men arbejdsindsatsen er naturligvis intensiveret i denne sidste fase.

7.5 Anvendte værktøjer

Udviklingen af projekt MAOV bygger på erfaringerne fra datamatiker-studiet på Lyngby Uddannelses Center fra sommeren 1998 til sommeren år 2000. Der er udvalgt de metoder blandt de oplærte som har relevans for projektets udførelse, og brugt til at sikre at projektet er udviklet med et tilfredsstillende resultat.

7.5.1 Projektstyring

Denne del bygger hovedsagligt på metoderne beskrevet af "Andreas Munk-Madsen³⁾" i værket "Strategisk Projektledelse". Der er oprettet en synlig indentitet for projektet ved at fastsætte navnet, give projektet en maskot og oprettet et web-site hvorpå man har kunnet følge udviklingen. Projektdeltagernes roller er afklaret. Projekforløbet er planlagt i en projektplan med faser og faselinjer, og projektet er styret derefter.

Mange af de metoder Andreas Munk-Madsen beskriver, er designet til at styre projekter der er væsentligt større og mere komplicerede end projekt MAOV. Der er derfor skåret kraftigt i de procedurer han beskriver, da det ville være spild af ressourcer. Det passer dog fint med hans påmindelser om at projektledelsen skal tilpasses det enkelte projekt.

Der har ikke været nogen kontrol af CMM, og om de forskellige kriterier har været overholdt. Erfaringer fra tidligere projekter har dog medført at forskellige former for kvalitetssikring er foretaget pr. intuition. Der har således været definition for dokument-standard og kode-standard. Der har været opstillet krav om godkendelse af dokumenter. Når gruppen kun er på to mand er det dog nemt at blive enige om sådanne specifikationer, som dog aldrig bliver nedfældet på papir.

Grundlæggende er der ikke brugt mange ressourcer på styring af den lille tætarbejdende gruppe. Om projektet har skredet i forhold til projektplanen har jævntligt været kontrolleret (et par gange om ugen), og hvis det har set ud til at der har været problemer, er der lagt kortvarige planer for at rette op på det.

7.5.2 System-udvikling

Denne del bygger primært på metoderne beskrevet af "Ivar Jacobson⁴⁾" og hans udviklingsmetode OOSE. Hans notation er dog ved at blive erstattet af UML, og det har vi fulgt hvor vi fandt det passende. OOSE specificerer de grundlæggende faser i udviklingen som projektet har fulgt og som rapporten er opbygget efter. Desuden har vi udarbejdet forskellige modeller fra OOSE for at beskrive systemet vi udvikler. Systemet er opdelt i interface/boundary-objekter, kontrol-objekter og entitets-objekter som Jacobson anbefaler, og der er holdt en synlig sporbarhed gennem faserne. Der er en generel beskrivelse af OOSE og dens faser i bilag 8.4 side 88.



7.5.3 Nye metoder

Den store udfordring for projektet var at lave et system der kørte på en WAP-telefon. Vi skulle implementere et system op imod Portal-to-Go som var en rimelig uklar størrelse ved projektets start. Desuden skulle vi lære at bruge XML, dog i et meget begrænset omfang i forhold til dets potentiale. Det var første gang vi har udviklede et system for en virksomhed, så det har været en prøvelse for de forskellige teorier vi har lært.

7.6 Kvalitetsstyring

Der har ikke været nogle faste retningslinjer for at sikre kvaliteten i projektet. Det skal dog ikke forstås sådan at kvalitetsstyringen har været svævende, tværtimod. Projektgruppens medlemmer har pr. definition et højt kvalitetsniveau, og er bl.a. sat sammen ud fra dette kriterie. Derudover har projektgruppen arbejdet sammen før på tidligere projekter, og kender derfor hinandens arbejdsmetoder.

Det har ikke været nødvendigt med en stram styring af projektet, da projektgruppen kun består af to medlemmer, som har arbejdet meget tæt sammen. Projektgruppens tætte samarbejde har bevirket at opståede uklarheder og uoverensstemmelser er blevet taget hånd om i tide, inden det evt. kunne have fået katastrofale konsekvenser for projektet. Såsom spildt arbejde og at projektplanen skrider, hvilket kan medføre brandslukning, der jo bestemt ikke er sundt for et projekt.

Det er endvidere meget ressourcekrævende at opretholde en stram kvalitetsstyring, og da gruppen, som sagt, kun består af to medlemmer er dette nedprioriteret. Begrundelsen for dette er det i forvejen meget høje kvalitetsniveau blandt gruppens medlemmer og det tætte samarbejde.

Internt i projektet er kvalitetsstyringen foregået på den måde, at hvert projektmedlem har gennemlæst og godkendt hvad den anden har udarbejdet, inden endelig godkendelse. Dette har været en naturlig del af gruppens daglige arbejde, og har betydet at projektets fokus/højekvalitet hele tiden er blevet holdt.

For at sikre sporbarheden gennem hele projektet og mellem de forskellige faser, har der været en løbende gennemgang af udarbejdede dokumenter og modeller. Dette har betydet at eventuelle uoverensstemmelser mellem dokumenter og modeller er blevet fanget tids nok til ikke at få indvirken på den videre udvikling, hvilket kunnet have kostet mange ressourcer, hvis der havde været arbejdet videre ud fra et forkert grundlag.

Eksternt er kvaliteten i projektet blevet styret ved afholdelse af reviews med vores vejleder Jørgen Karrebæk. Endvidere er et møde med andre projektgrupper, der arbejder med samme teknologier som os, blevet afholdt for at give et indblik i hvad de andre laver, og samtidigt virkede det som et forum hvor eventuelle mangler eller u hensigtsmæssigheder kunne opdages og tages til efterretning. Ved også at få nogle andre øjne på, var en rigtig god måde at sikre kvaliteten på.

I samarbejde med vores kontaktpersoner os Oracle Danmark, er kvaliteten også løbende blevet sikret ved afholdelse af møder og diverse uformelle møder. Dette har gjort at vi hele tiden har kunnet holde os på sporet, og unødigt tidspild er blevet undgået.



Vi synes selv at dette har været en meget fornuftig måde at sikre kvaliteten på, hvilket vores projekt da også er et bevis på. Nogle helt klare retningslinjer er dog blevet sat:

7.6.1 Dokument-standard

Alle dokumenter skrives i html-format, så de kan offentliggøres på projektets web-site. Der tillades kun simple html-tags: P, BR, UL, OL, LI, H1, H3, IMG, TABLE, TR, TD. Alle effekter lægges på i stylesheet, så der ikke skal være argumenter i de enkelte tags. Layoutet til papir-versionen lægges på i den afsluttende fase, og vil følge det specificerede stylesheet. Der oprettes en separat fil for hvert enkelt afsnit. Der oprettes en mappe for hver fase, og undermapper oprettes efter behov, løbende.

Der tages backup jævnligt i form af en zip-fil for hele projekt-kataloget. Backuppen flyttes til en anden computer.

7.6.2 Source-code standard

Følgende regler skal overholdes når der programmeres, for at opnå en ensartet source-code.

- Programmeringen holdes i Java.
- Alle klasser gemmes i en package ved navn: "oracle.dk.maov".
- Der skrives kommentarer så de passer til JavaDoc.
- Kommentarer skrives på dansk.
- Alle klasser, metoder og variable navngives på engelsk.

7.7 Dagbog

I dagbogen skrives hovedpunkterne for hvad vi har lavet den enkelte dag, i oversigtsform. Der vil altid blive brugt en del tid på rapportskrivning, eksperimentering og andre småting som ikke vil fremgå af dagbogen.

Uge 33 14.08.2000 - 18.08.2000

Mandag - Første dag. Sluset ind og vist rundt. Påbegyndte skærsilden.

Tirsdag - Afgangsprojekts-intromøde på skolen.

Onsdag - Interviewede folk i Oracle til Skærsilden.

Torsdag - Sidste interviews og layout til skærsild-rapporten.

Fredag - Sidste hånd på skærsilden, og projekt-etablering.

Uge 34 21.08.2000 - 25.08.2000

Mandag - Snakket med Søren om bruger-interviews og forberedt dem.

Tirsdag - Da vi ikke kunne snakke med brugere, studerede vi XML.

Onsdag - Studerede lidt Portal-to-Go, og udvekslede erfaringer med anden PtG-gruppe.

Torsdag - Første kontakt med brugerne. Studeret lidt mere PtG.

Fredag - Udarbejdet MAOV info-web-site, og sendt henvendelse til brugerne.

Uge 35 28.08.2000 - 01.09.2000

Mandag - Bruger-interview med Dan Kristensen, og tech-møde med Asger Jensen.

Tirsdag - Bruger-interview med Henrik Bustrup, bearbejdet forløbige krav.

Onsdag - Bruger-interview/IT-strategimøde med Kristian Bak, og ØMU-foredrag af Poul Nystrup.

Torsdag - Udviklings-miljø sat op af Asger, og studeret PtG-implementationsguide.

Fredag - Sempel programmeringsforsøg op imod PtG.



Uge 36 04.09.2000 - 08.09.2000

Mandag - Flere forsøg op imod PtG.
Tirsdag - PtG-prototype med forms oppe at køre.
Onsdag - PtG-prototype med forms og DB-tilgang oppe og spille!.
Torsdag - Interviews med tre sælgere: C.Fabricius, M.S.Christiansen & P.M.Jensen.
Fredag - Tidlig usecase-, PDOM-modeller.

Uge 37 11.09.2000 - 15.09.2000

Mandag - Mock-ups, og forberedelse af funktions-forslag til mødet dagen efter.
Tirsdag - Teknisk afgrænsningsmøde med Asger og Søren, og beskrevet afgrænsningerne.
Onsdag - Beskrevet PtG, og udarbejdet nye PDOM- og UseCase-modeller.
Torsdag - Udarbejdet kravmodel intro, UseCase-model-, PDOM- og mock-up-beskrivelser.
Fredag - Kravmodel færdiggjort, og gennemlæst internt.

Uge 38 18.09.2000 - 22.09.2000

Mandag - Udarbejdet ICE-model udkast.
Tirsdag - Diskuterede diverse designovervejelser, og lavede mock-ups over menustrukturen i MAOV.
Onsdag - Halvdelen af dagen gik med at flytte lokale, samt vi gik tidligt hjem.
Torsdag - ICE-model, og objekt-designovervejelser færdige.
Fredag - Møde med vejleder. Var nødsaget til kun at arbejde en halv dag.

Uge 39 25.09.2000 - 29.09.2000

Mandag - Opfølgning på vejledermøde. Start på implementering af medarbejder-funktion.
Tirsdag - Videre med implementering af medarbejder-funktion.
Onsdag - Diverse designovervejelser.
Torsdag - Tilrettet ICE-model. Start på analyse af CDM-database tabellerne.
Fredag - Udarbejdet navigations-diagram. Diverse smårettelser.

Uge 40 02.10.2000 - 06.10.2000

Mandag - Forberedelse af fremlæggelse til gruppemødet. Gruppemøde på skolen om eftermiddagen.
Tirsdag - Forsat analyse af CDM-database tabellerne.
Onsdag - Udarbejdet beskrivelse af Oracle's MAOVPrototype og PtG-XML dokumentation.
Torsdag - Møde med Rasmus Mencke om CDM-database tabellerne. Implementering af kontakt-funktion.
Fredag - Implementering af aktivitets-funktion.

Uge 41 09.10.2000 - 13.10.2000

Mandag - Serveren var nede, så vi kunne ikke programmere. Har istedet udarbejdet beskrivelse af teknologi-valg og databaser.
Tirsdag - Implementering af TAR-funktion og senest søgte kunder funktion.
Onsdag - Udarbejdet første version af SQLMachine og JavalnstantTester.
Torsdag - Implementering af senest søgte medarbejdere funktion og fejlretning af Aktivitets-funktion.
Fredag - Implementering af SendActionPoint- og SendEMail-funktion.

Uge 42 16.10.2000 - 20.10.2000

Mandag - Udarbejdet beskrivelse af round-trip, Oracle Danmark og forslag til eksterne funktioner.
Tirsdag - Første brugertest af MAOV, snak med Bo Nielsen om TAR-databasen og gennemgang af testresultater.
Onsdag - Udarbejdet beskrivelse af OOSE og kodemanøvrering.
Torsdag - Implementering af Prisliste-funktion, fejlretning af TAR-funktion og videre med implementeringen af SendActionPoint-funktion.
Fredag - Rettet sidste detaljer og fejl på MAOV, bl.a max linjer pr. side, tilbageknapper.



Uge 43 23.10.2000 - 27.10.2000

Mandag - Udarbejdet beskrivelse af SQL-kommandoer (SELECT's) og tilrettet kodemanøvrering beskrivelsen.

Tirsdag - Skrevet samtlige kommentarer til JavaDoc i koden.

Onsdag - Udarbejdet beskrivelse af layers design pattern, muligheden for ændringer i virksomheden, hvordan ændres systemet og tilrettet beskrivelsen af OOSE.

Torsdag - Beskrivelse af hvordan ændres systemet, start på sekvensdiagrammer og udarbejdede PDOM over hvordan systemet faktisk kom til at se ud.

Fredag - Beskrivelse af egne tabeller i MAOV databasen, færdig med sekvensdiagrammer, beskrivelse af input-argumenter, samt lavet første udkast af rapporten i word.

Uge 44 30.10.2000 - 03.11.2000

Mandag - Rettet de fejl fundet under gennemlæsning af det første udkast af rapporten.

Tirsdag - Beskrevet det meste af test-fasen, en masse om databasen og hvordan valgte kunder/medarbejdere gemmes.

Onsdag - Diverse småting rettet, testmodellen færdig og rettet tre bugs i koden.

Torsdag - Dokumenteret dele af processen vi har gjort brug af under udarbejdelsen af projektet.

Fredag - Arbejdet videre på mangler i rapporten, og sammensat gennemlæsningsversion 2 i MS-WORD.

Uge 45 06.10.2000 - 10.10.2000

Mandag - Rettet de fejl fundet under gennemlæsning af det andet udkast af rapporten.

Tirsdag - Lagt sidste hånd på rapporten, og fremstillet de 8 eksemplare.

Onsdag - Velfortjent biograftur.

Torsdag - **Aflevering !!!**

Fredag - Velfortjent drukatur :-)

7.8 Konklusion

Projektet er udført, og de opstillede mål er nået. Arbejds-processen og styringen af projektet har derfor været tilstrækkelig. Der har dog ikke været de store kriser eller problemer, og derfor ikke været brug for store indgreb styringsmæssigt. Der var et tidsproblem ved faselinjen imellem analyse og konstruktion, med det blev løst med en tilretning af projektplanen.

Det er generelt et problem at man ikke bliver færdig med de forskellige faser, men i dette projekt er der flere gange effektivt blevet lagt låg på de forskellige faser, når tilfredsstillende resultater er opnået. Dette bygger i høj grad på erfaringer fra tidligere projekter. Ligeså med kvalitetsstyringen, som også primært bygger på erfaringer. Den lille projektgruppe har gjort det nemt at blive enige om standarder, og projektet er derfor blevet meget ensartet udført.

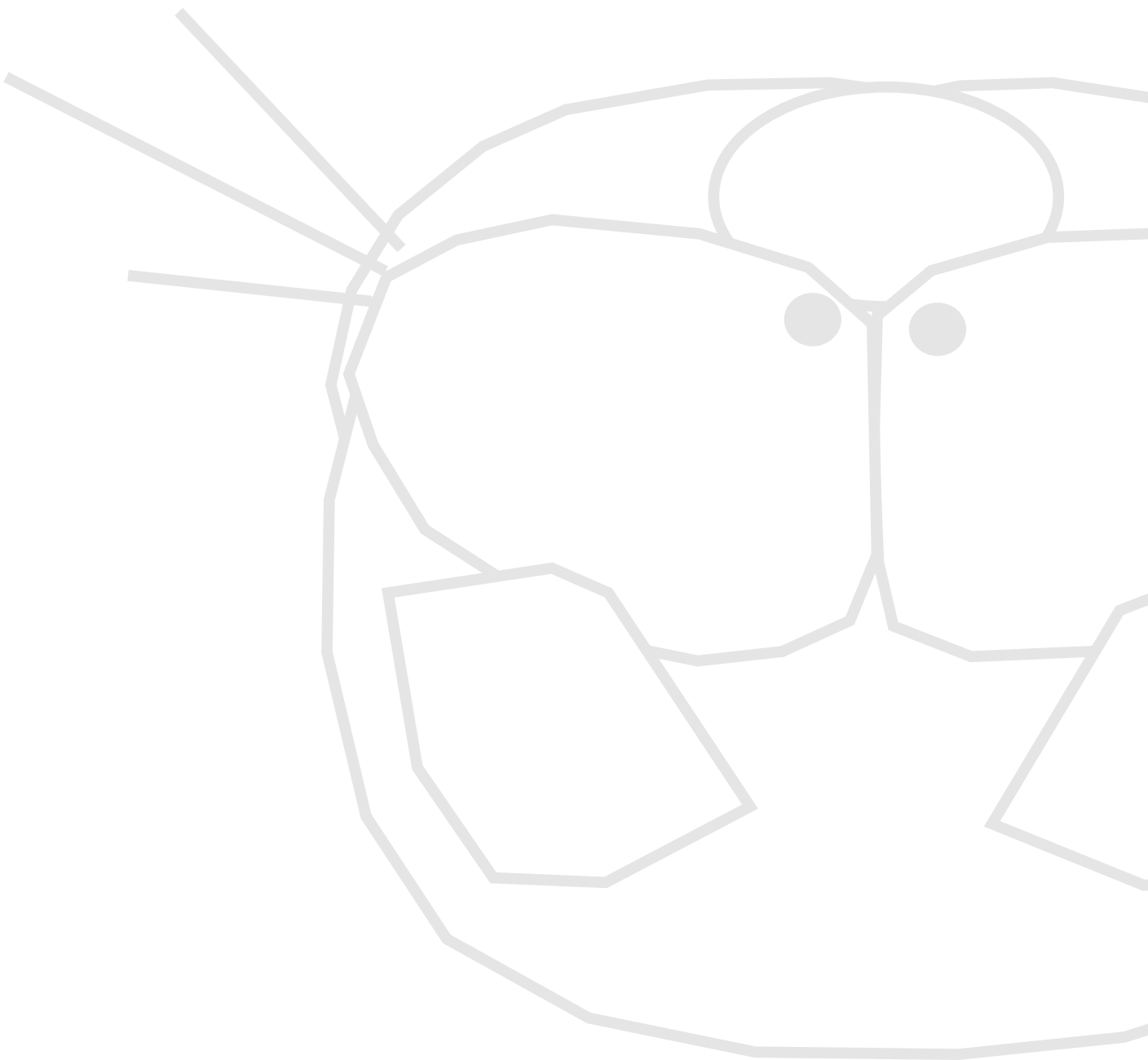
Når et projekt har kørt stort set uden problemer, tyder det på en tilstrækkelig styring, høj kompetance blandt projekt-deltagerne, en stor portion held, eller et simpelt forudsigeligt projekt. Det er nok en kombination af de fire. Den simple og lettere uformelle styring har passer fint til den lille tætarbejdende projektgruppe, men ville ikke fungere i et større projekt med flere mennesker indblandet.

De personlige forventninger for projektets deltagere er ligeledes blevet opfyldt. Der har været et godt samarbejde i projekt-gruppen og der har været tilfredsstillende support fra Oracle DK.



8. Bilag

Dette kapitel består af bilagene til rapporten. De kan læses uafhængigt af hinanden og resten af rapporten. Der bliver dog henvist til dem, de relevante steder i rapporten.



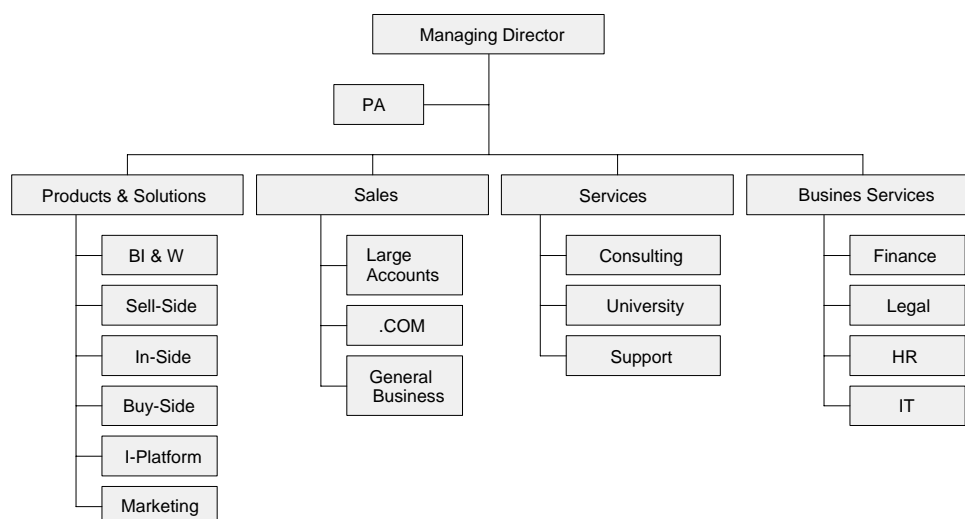


8.1 Oracle Danmark's afdelinger

I dette afsnit vil Oracle Danmark og dets afdelinger kort blive beskrevet, for at give et overblik over virksomheden vi har lavet vores afgangprojekt for og i.

Oracle Danmark er en del af det verdensomspændende IT-firma Oracle Corp., som er verdens næststørste leverandør af software og førende udbyder af e-business løsninger til Internethandel og webbaserede applikationer. Oracle Corp. har i alt 43.000 ansatte, heraf arbejder 400 i Danmark, og en samlet omsætning på 9,1\$ mia. dollar.

Oracle Danmark består af fire hovedafdelinger, som igen er opdelt i nogle underafdelinger (hovedområder). Se organisationsdiagram.



8.1.1 Products & Solutions

Products & Solutions har ansvar for salg og markedsføring af databaser og udviklingsværktøjer, såsom Oracle8, JDeveloper, Designer, case-værktøj, WebDB (kun et tool) og Video Server. Vi var tilknyttet underafdelingen I-Platform.

- **BI & W:** Forretningssystemer og datavarehuse.
- **Sell-Side:** Produkter kunden skal bruge for at sælge sine egne produkter.
- **In-Side:** Produkter kunden skal bruge internt i sin virksomhed for at styre den.
- **Buy-Side:** Produkter kunden skal bruge til at købe ind fra sine leverandører.
- **I-Platform:** Oracle's applikationsservere.
- **Marketing:** Tager sig af al marketing af Oracle's samtlige produkter, og står for alle aktiviteter såsom nyhedsbreve og seminarer, som Oracle tilbyder.



8.1.2 Sales

Sales står for alt salgsarbejdet.

- **Large Accounts:** Tager sig af kendte storkunder.
- **.COM:** Tager sig af alle .COM kunder.
- **General Business:** Tager sig af de kunder der er tilbage.

8.1.3 Services

Services står for al service der har med kunder at gøre.

- **Consulting:** Overtager kunderne efter salgs-afdelingen, og hjælper dem bl.a med at implementere deres nyindkøbte applikation. Desuden kan de hjælpe med at indføre CMM og SCM. Endvidere tager de sig af kunden hvis support ikke kan klare en kundes problem.
- **University:** Står for uddannelse af kunder.
- **Support:** Står for support af alle kunder, kunden kontakter support hvis et problem med Oracle's produkter opstår.

8.1.4 Business Services

Business Services står for al service internt i huset og rapportering af Oracle Danmarks forretning udadtil.

- **Finance:** Laver en del ledelsesrapportering og budgetering. Medvirker i udarbejdelsen af eksterne og interne regnskaber.
- **Legal:** Sørger for at der ikke sker snyd og bedrag.
- **HR:** I personaleafdelingen ordnes alt hvad der har med medarbejderne at gøre. De sørger for at nyansatte bliver sluset ind i systemet, og melder folk til uddannelser, rekrutterer, og registerer når folk flytter afdeling. Desuden står de for lønnen, og holder øje med om folk har været syge. HR står som bekendt for Human Ressource.
- **IT:** Hjælper alle de ansatte med tekniske problemer vedrørende deres arbejdsstationer, typisk software problemer.



8.2 Muligheden for ændringer i virksomheden

Under analysen af systemet, kom det frem at MAOV-systemet ikke må udvikles ifølge firmaet's globale IT-strategi. Det er der dog delte meninger om i de forskellige afdelinger.

8.2.1 Oracle's globale IT-strategi

Oracle er en meget stor virksomhed, med afdelinger i mange lande. Disse afdelinger har gennem lang tid udviklet deres egne IT-systemer til styring af virksomheden. Da det er nogenlunde de samme systemer de har brug for, var det usmart at de hver især brugte ressourcer på at udvikle de samme systemer. Prisen for udvikling af systemerne blev på den måde betalt flere gange. Tilmed blev systemerne ukompatible med hinanden, så der opstod en masse problemer med at synkronisere systemerne over lande-grænserne.

Det er Oracle blevet trætte af, så derfor vil de centralisere al udvikling på et sted (USA), som så laver generelle løsninger som kan bruges i alle afdelinger i hele verden. De er så fra start af lavet til at kunne kommunikere sammen, så hele den verdensomspændende virksomhed hænger sammen. Det bliver nemmere at vedligeholde og opgradere systemerne, og ingen lande halter bagefter med gammel software som de ikke har råd til at opgradere med egen nyudvikling.

Ideen er god, og der spares en masse penge. Men det betyder at der ikke må udvikles noget udenfor USA, og det betyder at de enkelte lande kan komme til at vente længe på at få udviklet et system de står og mangler, hvis det ikke er så højt prioriteret globalt som lokalt.

Det betyder for MAOV-systemet at det skulle være udviklet i USA, og lavet så generelt at alle lande ville kunne bruge det. Selvom det alligevel bliver lavet her i Danmark, må det ikke tages i anvendelse, da det ikke må vedligeholdes.

8.2.2 De ansattes syn på sagen

Holdningen hos de ansatte og deriblandt vores kontaktpersoner, var dog noget anderledes. De var glade for at få udviklet et system der kan hjælpe dem i hverdagen. Desuden ser de systemet som en god demonstration af mulighederne med Portal-to-Go overfor kunderne. Systemet er et supplement til de ansattes muligheder for informations-søgning, og bliver derfor ikke presset ned over hovederne på nogen. Derfor er der ingen af de ansatte der har noget imod systemet.

Vi har ikke kunne imødegå samtlige ansattes ønsker til systemet, men eftersom systemet kører via Portal-to-Go kan andre services tilføjes, så de kører parallelt med MAOV. For brugeren ser det så ud som om at alle services er et stort samlet system. Portal-to-Go er en portal, og MAOV-systemet er bare een service. Der har således været rygter om at Oracles kalender-system kommer som service til Portal-to-Go. En service der ikke var adgang til at implementere i MAOV, men den kommer så måske alligevel til brugerne.

8.2.3 MAOV som uofficielt system

De to modstridende holdninger til om MAOV må bruges, betyder er der måtte findes et kompromis. Systemet bliver sat i funktion, men ikke officielt. De ansatte kan bruge det, men der vil ikke være officiel support på systemet.



De ansatte satser på at når først systemet er oppe at køre, og folk bruger det, kan man bedre overbevise ledelsen om at det er værdifuldt, og dermed mindske deres modstand imod systemet.

Endelig kan man håbe at MAOV-systemet får status som pioner-arbejder der kan hjælpe de amerikanske udviklere med at lave en version der kan bruges globalt. Desværre er dokumentationen på dansk, så de må få en dansk udvikler med på deres amerikanske udvikler-team.

Hvorom alting er, så har systemet ikke kostet Oracle mange penge at udvikle. De har dog den fulde rettighed til det, og kan gøre med det hvad de vil.

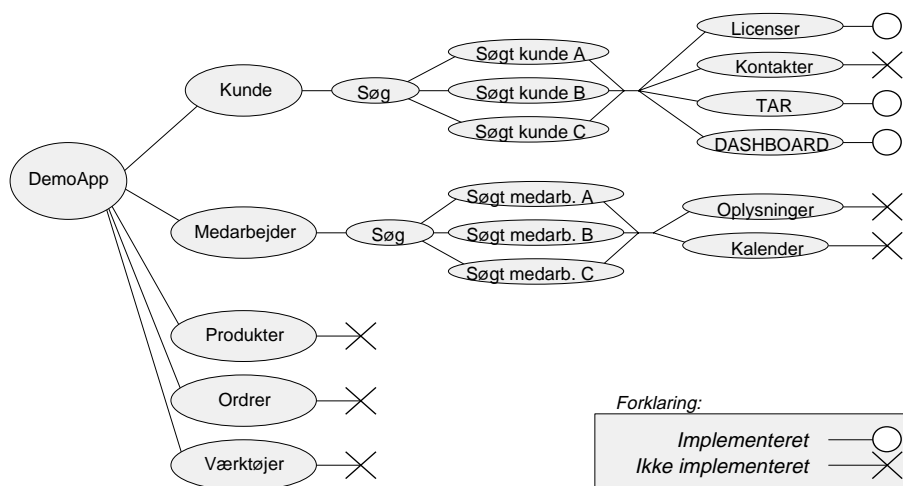


8.3 Oracle's MAOV-prototype

Oracle havde lavet en prototype af MAOV-systemet allerede før vi gik igang. Det var dog en meget begrænset udgave, og Oracle havde ikke tid til, selv at udvikle videre på den. Det er altså ideen til den gamle prototype der lægger grundlaget for MAOV, og det er også prototypens udviklere der er vores kontakt-personer hos Oracle.

Selvom prototypen er idé-grundlag for MAOV, har vi ikke genbrugt noget af systemet. Den var desuden udviklet i PL/SQL, hvorimod MAOV er udviklet i Java. Menu-strukturen har vi også valgt at redesigne, da der ikke er nogen dokumentation for, at den i prototypen er optimal.

Prototypen har en fuld menu-struktur, men det er et fåtal af menu-punkterne der er implementeret. Den er primært blevet lavet for at teste teknologien med Portal-to-Go af, og for at have en demonstration, som kan fremvises til kunder. Den fulde menu-struktur er som følger:



Selvom menu-strukturen i MAOV er udviklet uafhængigt af prototypen, er der visse logiske ligheder. Begge systemer er delt op efter om det er en kunde, en medarbejder eller noget tredje man vil finde oplysninger om. Man finder eksempelvis kunden først, hvorefter man kan se en masse oplysninger om den kunde, frem for først at skulle vælge hvilken oplysning man vil have, og derefter finde kunden frem. Prototypen er ikke på nogen måde tilpasset den enkelte bruger, som det er tiltænkt med MAOV. Prototypen ved ikke noget om, eksempelvis hvilke kunder en bruger har, så på prototypen skal man søge kunden blandt samtlige af Oracle's kunder.

Da prototypen henter en del oplysninger fra forskellige databaser, har vi ud fra den fundet mange af de forskellige oplysninger og hvor de ligger. MAOV skal dog bruge yderligere oplysninger end dem der er med i prototypen, som skulle findes ad anden vej.

Flere af Oracles medarbejdere ser gerne at prototypen, og dermed MAOV blev færdig. Prototypen er bare en prototype, og ikke ment som et endeligt system. MAOV derimod, har været igennem en komplet dokumenteret system-udvikling, og bør derfor bedre ramme brugernes behov.



8.4 Generelt om OOSE

Vi har i udarbejdelsen af dette afgangsprøve gjort brug af den objektorienterede systemudviklingsmetode OOSE, som er skabt af Ivar Jacobson⁴⁾. Begrundelsen for netop at benytte OOSE som udviklingsmetode hænger sammen med vores store erfaring med OOSE, samt at systemet skal implementeres i et objektorienteret sprog, hvilket jo lægger op til brug af en objektorienteret udviklingsmetode. Udover dette så blev der heller ikke stillet noget krav fra Oracle's side om at vi skulle benytte en specifik udviklingsmetode.

Hvorfor overhovedet benytte en systemudviklingsmetode, kan man ikke bare skrive sit program direkte ud fra kravene. Nej, det kan simpelthen ikke lade sig gøre, måske hvis det er et meget lille program. Men når man snakker om systemer af en vis størrelse, så er man nødt til at benytte en eller anden form for systemudviklingsmetode, da det at udvikle sådanne systemer er en meget kompleks opgave. Målet er at få et pålideligt system, der udfører sine opgaver rigtigt.

Systemudvikling er typisk en så kompleks opgave at det ikke kan lade sig gøre at have alle bolde i luften på en gang. Man er derfor nødt til at håndtere kompleksiteten på en eller anden struktureret måde. Det kan man gøre ved at benytte en systemudviklingsmetode, som indeholder forskellige modeller der hver fokuserer på forskellige dele af systemet. På denne måde deler man kompleksiteten op i mindre dele, og derved kan man styre hele systemets kompleksitet. OOSE består af fem sådanne modeller, som indgår i OOSE's tre faser:

- Analysefasen (krav- og analysemodel)
- Konstruktionsfasen (design- og implementationsmodel)
- Testfasen (testmodel)

Intensionerne i OOSE er bl.a. at: nedsætte livscyklus omkostninger ifb.m. håndtering af edb-baserede systemer (dvs. gøre vedligeholdelse lettere), give sporbarhed, give indkapsling (dvs. ændringer er så lokale som mulige), give lille semantisk spring (dvs. lille afstand mellem den fysiske verden og den måde vi repræsenterer den fysiske verden i en computer), og sørge for at udviklede dele kan genbruges.

I det følgende vil OOSE, dets faser og modeller blive beskrevet.

8.4.1 Analysefasen

Analysefasen skal danne grundlag for kravene til systemet, samt give systemet en holdbar/robust objekt struktur der nemt kan ændres. Der bliver lagt vægt på at interagere med brugerne.

Analysefasen indeholder to modeller:

- Kravmodel
- Analysemodel



8.4.2 Kravmodel

Kravmodellen skal bruges til at afdække de funktionelle krav til systemet, set fra brugerens perspektiv. Denne model udvikles oftest i tæt samarbejde med brugerne. Kravmodellen indeholder tre modeller:

- Use-case model
- Domæne Model (tidligere kaldt PDOM)
- Grænseflade model (Mock-up's)

Use-case model bruges til at vise hvad systemet skal kunne overfor brugeren. Er en beskrivelse af hvordan skal vi ved hjælp af edb udfører dit (brugerens) arbejde. Dvs. en beskrivelse af arbejdsprocesser.

Aktører (roller man spiller): hvordan bruger de forskellige aktører systemet.

Domæne Model: er de centraler begreber (objekter/entitetsobjekter) i systemet, ofte navneord, hvordan data gemmes/hvilke data systemet omhandler. Typisk tegne arv (statiske relationer). Om Domæne Modellen skal beskrive/indeholde generelle udsagn om data eller helt ned på primitiv dataform, det kan man diskutere meget om. Syntaks: OMT(UML syntaks). Sikre at vi taler samme sprog.

Grænseflade model (Mock-up's): Er en tegning af hvordan systemet fremtræder for brugeren. Det er nemmere for brugeren at forstå, end at læse en beskrivelse.

8.4.3 Analysemodel

Hvordan kan systemet ideelt modelleres. Laves ud fra den holdning at alt kan lade sig gøre (den ideelle verden). Der er to grunde til dette. For det første er det meget nemmere at arbejde under ideelle forhold: det gør nemlig at man kan minske kompleksiteten og herved rette alle sine kræfter mod at gøre systemets opbygning mere stabilt, robust og logisk. For det andet så vil teknologien, om vi kan lide det eller ej, ændre sig i løbet af systemets levetid, og derfor er det ikke hensigtsmæssigt at lade den nuværende teknologi påvirke systemets opbygning. Analysemodellen indeholder to modeller:

- ICE model
- Udvidet use-case model

ICE model: Usecase-model og Domæne Model inddrages. Hvergang der er en overskridelse af grænsen: lav grænsefladeobjekt (præsenterer systemet for brugeren), kontrolobjekt (kører processen igennem/spilfordeler), styrer usecasen, er primært metodebærende) og entitetsobjekter (databærende). Denne model kaldes også for ICE-modellen (I=interface, C=control, E=entitit)

En klasse består af noget data (entitetsobjekter/modelsiden, det der skal huskes), metoder(grænsefladeobjektet) og kroppen (er kontrolobjektet). Det er hele grundlaget for hele objekt teorien, faktisk ICE-modellen.

Udvidet use-case model: Hvis det samme står i to usecase-modeller, slås de sammen, og lav uses-relation, er det der altid skal udføres (eks. login). Hvis der står, hvis dette så gør, eller hvis noget andet så gør dette, så lav extends-relation, det er det der sommetider skal udføres (eks. menu-valg). Det er svært at lave, så derfor laves det kun i interne usecase-beskrivelser (det er ikke fair overfor brugeren, at vise dem dette). Dvs. når de bliver så store.



8.4.4 Konstruktionsfasen

Det er nu at systemet bliver modelleret ud fra den teknologi det skal konstrueres ud fra.

Konstruktionsfasen indeholder to modeller:

- Designmodel
- Implementationsmodel

8.4.5 Designmodel

Transformere analysemodel fra ideel verden til teknologisk verden, dvs. til den teknologi systemet skal implementeres i. Analyseobjekter bliver til designblokke (mange objekter).

8.4.6 Implementationsmodel

Transformere designmodel til kildekode i givet programmeringssprog. Fra Domæne Model har vi datasiden af de centrale objekter (entitetsobjekterne). Fra sekvensdiagrammerne har vi klassenavnene og metode hoveder til objekterne, dele af metodekrop i form af pseudokode. Så nu er det bare at kode.

8.4.7 Testfasen

I denne fase skal det udviklede system stå sin prøve, dvs. kan det udføre det som der er opstillet i kravmodellen.

Testfasen indeholder een model:

- Testmodel

8.4.8 Testmodel

Testmodellen laves for at verificerer det udviklede system. Indeholder hovedsageligt test specifikationer og test resultater.



8.5 Round-trip

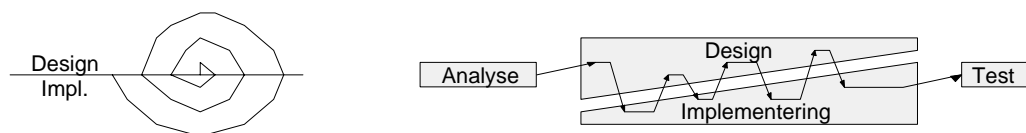
Round-trip går i sin enkelthed ud på at man designer og implementerer sit system sideløbende, som alternativ til at designe systemet fuldt ud, og implementere bagefter.

Det har tidligere været forventet at et system er designet og dokumenteret fuldt ud, før man begynder at implementere. Det er nødvendigt hvis det er et stort system, som skal deles ud til mange udviklere, og de skal være enige om hvad de laver. Project MAOV derimod er lille, og udvikles af kun to personer der arbejder tæt sammen, så der er ikke nogen kommunikationskløft når man er i tvivl om hvordan noget skal laves.

Det har vist sig at det er meget svært at designe alle detaljerne inden man implementerer, da man på det tidspunkt ikke kender alle detaljerne og begrænsningerne/mulighederne i teknologien. Derfor er det smartere at lave et round-trip hvor man designer lidt, implementerer lidt, designer lidt, implementerer lidt o.s.v.

Typisk vil man i starten designe systemet i grove træk, for at få et ensrettet grundlæggende design. Det har vi gjort i afsnittet "designovervejelser". Derefter laver man en vertikal prototyping ud fra designovervejelserne, for at se om de holder. Der vil typisk være rettelser til design-retningslinierne, efter de er forsøgt anvendt første gang, og det er vigtigt at få rettelserne lagt på plads. Med nogle gode retningslinier, kan man gøre implementeringen væsentligt nemmere.

Med round-trip bygger man mere og mere på systemet, og designer detaljerne når man når til dem. Det minder unægteligt meget om spiral-modellen for udviklingsparadigmer, men i tilfældet med roundtrip vil analysefasen grundlæggende være afsluttet. På den måde sikrer man at projektet ikke drejer i en forkert retning pga. teknologiske muligheder, men holder sig til det system man har analyseret sig frem til.



De to illustrationer skal vise det samme, at man skifter mellem design og implementering flere gange. Spiralen illustrerer at systemet bliver større og større, jo længere man kommer, mens den anden model viser skiftene i forhold til tiden på en tidsplan. Man vil starte med design, skifte nogle gange for at ende med at implementere det sidste. Som illustrationen viser vil design-delen være størst i starten, for at udfases og implementeringen bliver den dominerende.

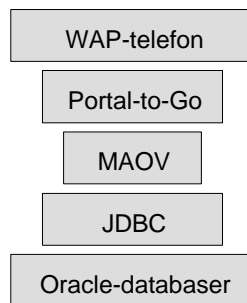
Round-trip er en mere løs styring, og man kan nemt komme til at glemme at dokumentere designet undervejs. Det skal man være opmærksom på. Der er dog ikke den store forskel på roundtrip og udvikling med faste faser med lidt iteration, så i realiteten er det bare et nyt navn for det man hele tiden har gjort.



8.6 Design-pattern: Layers

MAOV-systemet følger et design-pattern der kaldes "layers" ^{1) 5)}. Det går ud på at systemet er opbygget af lag på forskellige abstraktions-niveauer, som udnytter hinanden, og på den måde gør de enkelte lags opgaver mere afgrænsede og overskuelige.

At MAOV er blevet lagdelt kommer sig i høj grad af at de systemer det er designet til er delt op i lag. Det øverste lag er WAP-telefonen (e.lign) og det nederste lag er Oracle's databaser med bl.a virksomhedsoplysninger. Der er langt imellem de to lag, men heldigvis er der standard-komponenter der kan lægges imellem de to ydre lag, så de mødes i midten i et Java-baseret lag, hvor MAOV kommer til at ligge. I den øvre ende er det Portal-to-Go der skaber bro mellem WAP-telefonen og Java, og i den nedre ende er det JDBC, der skaber bro imellem Oracle-databaserne og Java. I midten ligger så det rent Java-baserede MAOV-system, som sørger for at det er de rigtige data der kommer fra Oracle-databaserne til WAP-telefonen.



På denne grove model, er MAOV-systemet umiddelbart kun et lag, men det er i sig selv også delt op i flere lag, hvilket de andre lag i den grove model iøvrigt også er. Men lad os først se på specifikationen på et "layered" design-pattern.

8.6.1 Definition af layers

Et lagdelt system er et ordnet sæt af virtuelle verdener, som hver er bygget ud fra det underliggende lag, og leverer basis-funktionalitet for laget ovenover. Et lag kender kun laget lige under sig, dvs. det kender hverken lag længere nede, eller lag over sig.

Der er et alternativ til denne såkaldte "lukkede arkitektur", som kaldes "åben arkitektur". I den åbne er det tilladt for lag at bruge samtlige underliggende lag direkte. Det kan give en optimeret og mere kompakt kode, men gør lagene langt mere afhængige af at de andre lag ikke ændres. Man vil derfor oftest fortrække den lukkede.

Et velkendt eksempel på layers er netværks-protokoller der typisk ligger i lag, hvor de nederste er mest hardware-specifikke, og de øverste er applikationer som brugerne ser.

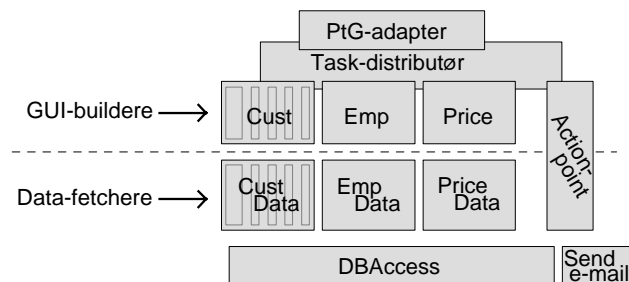
8.6.2 Kombination med subsystemer

I et større system vil det ikke altid være tilstrækkeligt at dele systemet op i vandrette lag. Man kan derfor kombinere opdelingen vandret med lodret opdeling i subsystemer. På den måde får man en slags blokdiagram over hvilke dele systemet består af, og hvilke der ligger over hvilke



8.6.3 MAOV-systemets lag

MAOV er primært delt op i to lag. Et lag der genererer XML-dokumenterne til brugerne (GUI-buildere), og et lag der henter de fornødne oplysninger fra databasen (Data-fetchere). Dertil kommer tre hjælpelag der henholdsvis skaber forbindelsen til Portal-to-Go, distribuerer opgaverne og skaber selve forbindelsen til databasen. De to primære lag er desuden delt op i tre subsystemer udfra deres afgrænsede funktion (Customer, Employee og Pricelist) plus et fjerde subsystem (action-point) som var tiltænkt at kunne støtte de tre andre. Customer-subsystemet er det mest komplekse, og er derfor yderligere delt op.



Hoved-delen af systemet er de 6 kasser i midten af modellen. De er delt ind vandret i de to lag, og lodret i de tre subsystemer. Subsystemerne går altså på tværs af lagene, eller sagt med andre ord, de tre subsystemer er ens opdelt i lag. De tre subsystemer fungerer nogenlunde ens, men bruger ikke hinanden. Action-point subsystemet er meget lille og derfor ikke delt i de to lag, som man kunne mene at det burde. Det bruger til gengæld både DBAccess og SendEmail som underliggende lag.

Denne opdeling minder meget om systemets opdelingen i klasser, som illustreret på ICE-modellen. Mange af klasserne, har her fået deres eget sub-system. "Task-distributør"-laget er dog lidt specielt, idet det er delt ud over flere klasser. Specifikt udgøres den af invoke-metoden som er nogenlunde ens i de forskellige objekter den er implementeret i, og derfor har en logisk funktion som sit eget lag. Det ligger dog reelt indbygget i de andre lag, derfor er den illustreret som overlappende.

Den primære idé med at dele systemet op de to lag var at adskille database-tilgangen fra den del af systemet der bygger siderne op som skal vises til brugeren. Den ene halvdel af systemet henter data, og den anden halvdel sender dem videre.

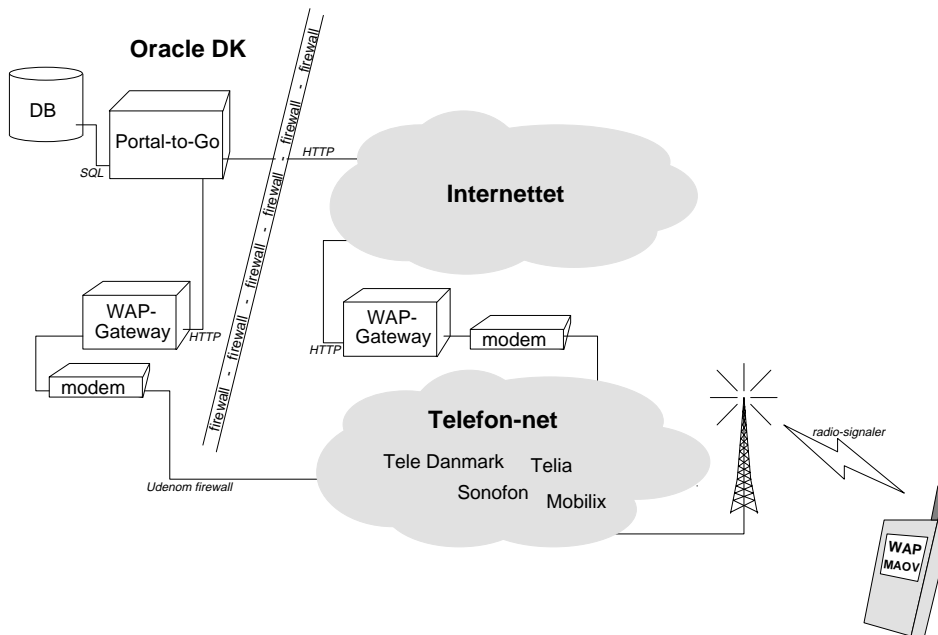
8.6.4 Følger MAOV så layer-patternet?

MAOV-systemet har ikke kun et lag-delt design, men det er en del af det samlede design, og det følger specifikationerne. De underliggende lag kender ikke de overliggende lag, men tilbyder den basis-funktionalitet de kræver. Det er en lukket arkitektur, da der ikke er nogen der springer lag over.



8.7 WAP-signalets rute

Illustrationen viser hvordan data kommer fra databasen, ud til WAP-telefonen. Der er umiddelbart to ruter signalerne kan tage. Enten gennem internettet eller direkte over telefonnettet til et firma's private WAP-gateway. På den sidstnævnte måde, kan man slippe udenom en firewall der beskytter virksomhedens intranet imod internettet. Det er kun medarbejdere der har adgang til at ringe til den private gateway, så uvedkommende kan ikke bruge systemet. Det er den rute MAOV kommer til at tage!



Der er mange måder at konfigurere systemet på, men typisk har mobiltelefon-udbyderen en WAP-gateway der giver adgang til internettet. På internettet kan hvem som helst sætte en WML-server op, da siderne sendes via HTTP - den samme protokol som bruges til hjemmesider. Der skal bare sendes WML i stedet for HTML. Reelt kan WAP-telefonen få fat i alle hjemmesider på internettet, men de små telefoner kan ikke vise dem på deres små display's. Derfor har man lavet det forsimplede sprog WML. Det betyder at hjemmesider skal omskrives til WML for at kunne vises på WAP-telefoner. Fordelen ved det er, at man så tvinger folk til at lave WML-sider hvor indholdet så bliver mere koncentreret.

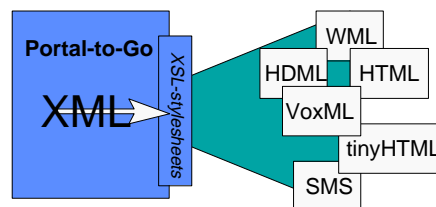
Fremtiden for mobilt internet varsler en masse ændringer. Rent teknisk vil mobil-nettet blive pakke-koblet, så man er online altid. Hastigheden vil stige markant, og mobiltelefonerne vil udvikle sig til små computere, med mange flere muligheder. I sidste ende vil de vel blive så avancerede at de kan vise HTML direkte, og WML vil blive overflødig. Men det ligger år ud i fremtiden, og indtil da må vi nøjes WAP...



8.8 Portal-to-Go - Hvad er det?

Mottoet for Portal-to-Go er: "Any service... ..on any device". Internettets anvendelse vil udvides fra kun at kunne bruges fra en internet-browser på computeren, til mange nye håndholdte enheder. Mange af dem kræver at hjemmesiderne omskrives/optimeres specielt til hver enkelt enhed. Portal-to-Go fra Oracle løser problemet, så man på en nem måde kan få vist de samme oplysninger på alverdens forskellige enheder, uden at skulle omskrive dem manuelt til hver enkelt.

Fundamentet i PtG er, at man har eet standard-sprog som kan konverteres til alle de andre sprog efter behov. Standard-sproget er XML, og det er i sig selv designet til at man kan konvertere XML-dokumenter til andre sprog, ved hjælp af stylesheets (XSL). Det er nøjagtig det PtG udnytter.

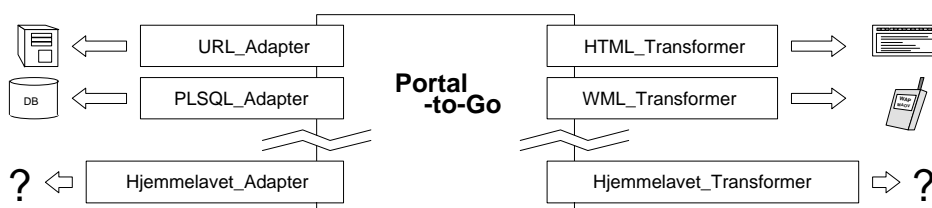


Portal-to-Go har indbygget stylesheets til stort set alle enheder, og det er simpelt at tilføje nye stylesheets, når nye sprog opstår. Man behøver derfor (teoretisk) ikke bekymre sig om hvilken maskine der skal vise det man laver i sidste ende, for det kan vises på hvad-som-helst.

Problemet ved et fælles sprog er, at det skal begrænses til kun at understøtte funktioner som kan vises på samtlige enheder. Man er derfor nødt til at sætte en mindste fællesnævner, så alle enhederne kan være med. Det er eksempelvis stærkt begrænset hvad man kan med en WAP-telefon i forhold til hvad man kan i en WEB-browser, og når begge dele skal understøttes, sætter WAP-telefonen grænsen for hvad der er muligt i PtG.

I mange tilfælde er de begrænsede muligheder ikke et problem. Man kan trods alt lave mange ting indenfor rammerne. Præsentationen i WEB-browseren behøver heller ikke at være så grim visuelt som WAP-versionen, for man kan sætte en masse flotte ting på HTML'en i stylesheetet.

Hvordan får man så sine data ind i PtG? XML-dokumenterne opbygges i adaptere. En adapter henter data fra diverse kilder og leverer dem til PtG som XML. Der er en række generelle standard-adaptere man kan bruge, og man har mulighed for at lave sine egne.





Hele PtG er implementeret i Java, og det er adapterne også. At lave sin egen adapter er derfor ikke sværere end at lave en java-klasse der arver fra en PtG-adapter, og som kan levere korrekt XML til PtG.

I vores tilfælde med projekt MAOV har vi lavet vores egen adapter, hvilket betyder at al udviklingen af MAOV-systemet kan holdes i Java. Vi behøver ikke bekymre os om det tekniske i brugerfladen, da PtG sørger for det hårde arbejde, men kan koncentrere os om brugervenligheden. Vores endelige applikation vil kunne vises og bruges på internettet, WAP-telefoner og fremtidige online enheder, uden at vi behøver at gøre noget for at konvertere til de forskellige enheder, og det er det der er hele ideen med Portal-to-Go.



8.8.1 WEB-stripperen

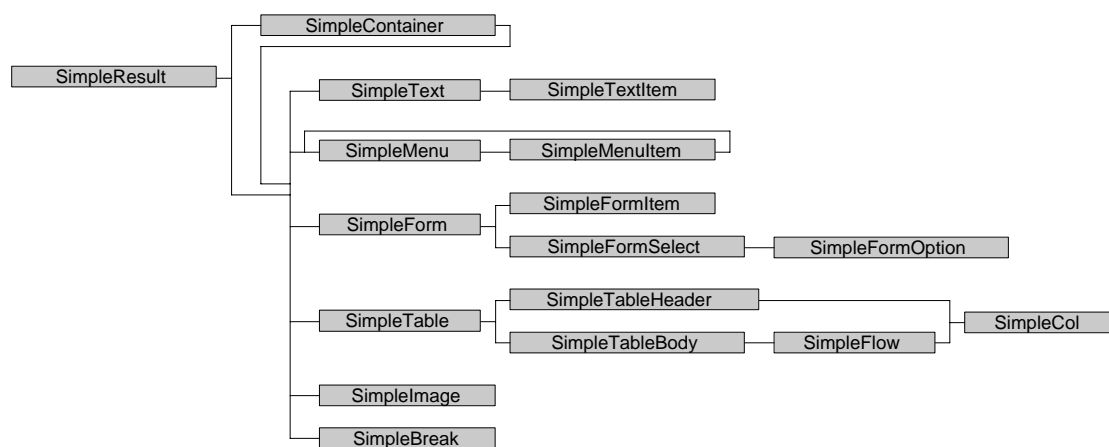
Portal-to-Go har en funktion der kaldes en WEB-stripper. Den bruges til at tage enkelte værdier fra web-sites. Det virker ved at man specificerer hvilket felt på sitet som værdien står i, og hver gang værdien så skal bruges, henter WEB-stripperen siden, og klipper den ønskede værdi ud. Det er ikke en særlig pæn måde at hente data på, for hvis sidens opbygning ændres, vil WEB-stripperen ikke længere kunne finde den ønskede værdi. WEB-stripperen bliver ikke brugt af MAOV-systemet.



8.9 Portal-to-Go's XML-standard

Portal-to-Go's primære funktion er at konvertere XML-dokumenter så de kan vises på en vilkårlig enhed. XML-dokumenter er dog en fællesbetegnelse for en masse forskellige typer dokumenter, og det er ikke muligt at kende alle typer. Portal-to-Go har derfor sin egen type XML-dokumenter med egne tags (DTD), og forstår kun dem.

Et PtG-XML-dokument består af en række grafiske elementer der skal vises på modtager-apparatet. Det kan eksempelvis være en tekst-stump, et indtastningsfelt, et billede eller en menu. Man bygger sin side op af disse simple objekter og PtG sørger for at de vises rigtigt på de forskellige enheder. Herunder vises de objekter/tags man kan bruge i et PtG-XML-dokument (DTD'en):



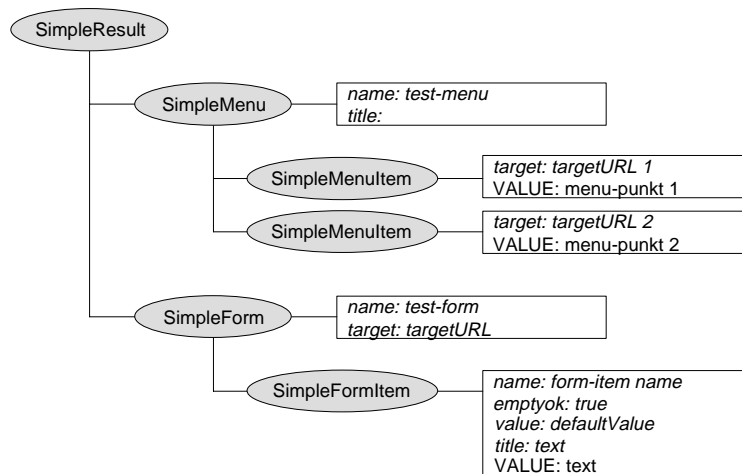
Nogle objekter skal være inde i andre. Eksempelvis skal et simpleMenuItem være i en simpleMenu. Hvis man placerer objekter forkert vil XML-dokumentet blive afvist.

Her er et lille eksempel på en side, vist som XML, som det hierarki af oplysninger XML'en beskriver, og en simpel oversættelse til HTML. Siden indeholder en menu med to punkter og et indtastningsfelt.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SimpleResult>
  <SimpleMenu name="test-menu" title="">
    <SimpleMenuItem target="targetURL 1">
      menu-punkt 1
    </SimpleMenuItem>
    <SimpleMenuItem target="targetURL 2">
      menu-punkt 2
    </SimpleMenuItem>
  </SimpleMenu>
  <SimpleForm name="test-form" target="targetURL">
    <SimpleFormItem name="form-item name" emptyok="true"
value="defaultValue" title="text">
      text
    </SimpleFormItem>
  </SimpleForm>
</SimpleResult>
```



XML-dokumentet beskriver en struktur af oplysninger. Der er et SimpleResult der indeholder en SimpleMenu og en SimpleForm, som igen indeholder nogle Items, som har nogle attributter. Det kan tegnes som en træstruktur af informationer.



Portal-to-Go konverterer så XML'en til det ønskede format, eksempelvis HTML. Der er dog mange måder man kan ønske sin HTML skal se ud, så dette er bare et simpelt eksempel. Man kunne i sit XSL tilføje en masse smarte visuelle effekter og billeder. Dette er altså bare et simpelt eksempel.

<pre>menu-punkt 1
 menu-punkt 2
 <FORM> <INPUT TYPE="text" VALUE="defaultValue"> <INPUT TYPE="submit" VALUE="OK"> </FORM></pre>	<p>menu-punkt 1 menu-punkt 2</p> <p>defaultValue OK</p>
---	---

Det kan umiddelbart virke besværligt, at siderne skal skrives i XML og konverteres til HTML via XSL. Men hvis man eksempelvis har 1000 sider, er det noget nemmere at tilføje et nyt stylesheet end at ændre på samtlige sider. Man vil samtidig kunne få vist siderne på den gamle måde, uden at skulle have begge versioner liggende, og optage plads. Siderne kan vises på eksempelvis 10 forskellige måder, men siderne ligger der kun een gang.

Det er standarderne XML og XSL, der gør det muligt at præsentere sider forskelligt. Portal-to-Go er bare en implementation af det system, der hjælper programmøren med noget af det hårde arbejde.



8.10 Det endelige system

Det endelige system som det blev implementeret kan kun bruges af Oracle DK's medarbejdere. Det er derfor ikke let for udefrakommende at se det endelige resultat, nemlig systemet. Her er derfor en række screenshots af systemet som det ser ud i sin endelige form. Her er ikke samtlige sider af systemet, men et repræsentativt udsnit.

Eftersom systemet kører gennem Portal-to-Go har det automatisk flere forskellige brugerflader. De samme situationer er vist, med tre forskellige brugerflader.

8.10.1 WML på WAP-emulator



Brugerfladen på WAP-telefonen er den der primært er udviklet systemet til. Det er den mest begrænsede brugerflade af de tre, og kan kun vise et lille udsnit af de enkelte sider ad gangen (der skal scrolles for at se hele siden). Der er ikke et normalt tastatur, så når man skal indtaste noget, eksempelvis login, vælger man feltet der skal skrives i, skærbilledet skifter til en side beregnet til at indtaste i med tal-tasterne. Det er meget besværligt at indtaste tekst. De to sidste sider der er vist i eksemplet er kun et lille udsnit af siderne. Man kan "scrolle" op og ned på siden for at se det hele.



8.10.2 HTML på "rm" (simpel HTML)

Name:
Password:

MyHome
Kunde
Medarbejder
Pris-liste

4K Beton
Mærsk Data AS
Brøndby Kommune, Brøndby
WM-Data A/S

Del af
navn

Brøndby Kommune
Park Alle 160
Brøndby
43282828

Licenser
Aktiviteter
TAR
Kontakter

Denne WEB-brugerflade er den mest simple. Den viser det næsten som på WAP-telefonen, med den forskel at man kan se meget mere af siden ad gangen. I eksemplet er browservinduet gjort meget lille, men det kan jo strækkes til at fylde hele skærmen. Eftersom den minder så meget om WAP-versionen er den god til at teste systemet med.

8.10.3 HTML på "PAPZ" (avanceret HTML)

ORACLE
Portal-to-Go

User Name:
Password:

Welcome MAOV Projektet
Nov 3, 2000

MAOV projektet
• Kunde
• Medarbejder
• Pris-liste

ORACLE
Portal-to-Go

Welcome MAOV Projektet
Nov 3, 2000

Brøndby Kommune, Brøndby
Dansegård AS
Kalenhamer Løftstuen, Kastrop
Telia Danmark AS, Ballerup
Nokia Danmark AS
Telia Telecom AB
Oracle Danmark AS
4K Beton
Mærsk Data AS
WM-Data AS

Del af navn

Welcome MAOV Projektet
Nov 3, 2000

Brøndby Kommune
Park Alle 160
Brøndby
43282828

Licenser
Aktiviteter
TAR
Kontakter
>> Hovedmenu <<



Dette er et eksempel på at systemet kan præsenteres på en mere avanceret måde, ved at benytte et mere avanceret stylesheet (XSL). Det skal illustrere muligheden for at tilpasse brugerfladen efter behov.

Sådan ser det endelige system altså ud. De viste eksempler er kun et lille udsnit af det samlede system, men de andre sider af systemet minder om de viste. Portal-to-Go sætter store begrænsninger for mulighederne for brugerfladen, så de forskellige sider er bygget op af de samme elementer.



8.11 Oprettelse af brugere til MAOV-systemet

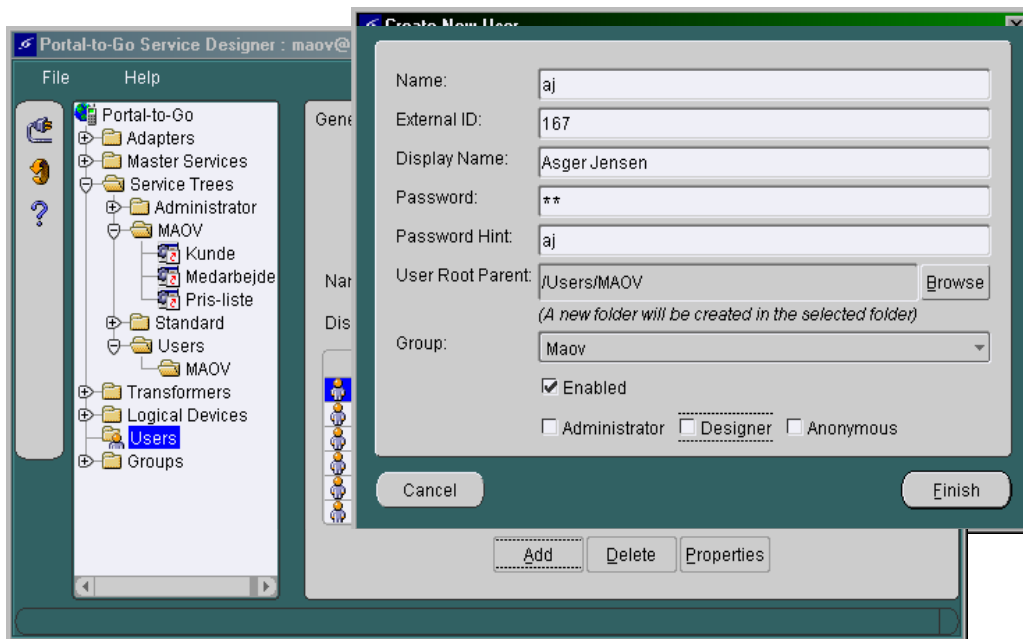
Når en ny bruger skal tilkobles til MAOV-systemet, så foregår det ved at bruge programmet Service Designer, som er en del af Portal to Go. Det er et krav at en ny bruger også eksisterer i medarbejdertabellen "hrv_people_info" i "prod" databasen.

Man går ind under "Users" for at tilføje en ny bruger se nedenstående screenshot. Her angives den nye brugers navn, login og password, endvidere skal et external ID angives. Dette ID bruger MAOV-systemet til at bestemme hvilken bruger der er logget på, således at bl.a brugersens seneste valgte kunder og medarbejdere kan findes frem.

Det er ikke lige meget hvilket external ID en bruger får. Det er et krav at den ny brugers external ID er det samme som det ID brugeren har i tabellen "hrv_people_info" i "prod" databasen. Dvs. at external ID skal være lig med attributten "person_id" i tabellen "hrv_people_info". Begrundelsen for at de skal være lig med hinanden, er at MAOV-systemet skal bruge brugerens navn og e-mail adresse når der sendes action points. Disse oplysninger findes i tabellen "hrv_people_info".

Brugeren skal tildeles en root-folder og der skal angives hvilken gruppe brugeren skal tilknyttes. Root-folder skal være "users\maov" og gruppen skal være "maov", for at kunne bruge MAOV-systemet.

Brugeren er nu oprettet, og MAOV-systemet er tilgængeligt.





8.12 Forslag til eksterne funktioner

MAOV-systemet er udelukkende udviklet med henblik på at gøre diverse interne funktioner mobile, men man kunne sagtens forestille sig nogle eksterne funktioner som det ville være relevant at udbygge MAOV-systemet med. Det kunne for eksempel være relevant med adgang til diverse trafikinformationer, eller kraks vejviser.

Det vil være forholdsvis enkelt at udbygge MAOV-systemet med diverse eksterne funktioner, hvis de i forvejen findes på internettet. For den teknologi, som MAOV-systemet benytter (Portal to Go), muliggør nemlig en nem integration imellem internettet og bl.a WAP.

Den mest enkelte eksterne funktion at integrerer med MAOV-systemet, er en funktion der udbydes som en WAP-tjeneste. I dette tilfælde er det simpelthen bare at oprette et link til denne funktion i MAOV-systemet.

Hvis funktionen ikke udbydes som en WAP-tjeneste, men findes på nettet, så kan man bruge Portal to Go's web-stripper til at "klippe" bestemte ting ud, som man godt vil have adgang til.

Denne funktionalitet, som Portal to Go tilbyder, gør at det med få udviklingsomkostninger er muligt at integrerer allerede eksisterende eksterne services/funktioner i MAOV-systemet nemt og hurtigt.

Herunder er en liste med forslag til nogle eksterne funktioner. De er ment som en inspiration og kan måske være med til at afstedkomme mange flere.

- **Trafikinformationer:** Det er vigtigt at være informeret om den aktuelle trafiksituation, når man er ude at køre. Det kan spare en meget tid i en forvejen travl hverdag.
- **Kraks vejviser:** Hurtig adgang til kørevejledning.
- **Aktiekurser:** En "nice to have" funktion, man kan følge nogle aktier man selv vælger.



8.13 Møde-referater

Referater af alle afholdte møder er at finde i det følgende. De er sorteret i kronologisk rækkefølge.

21.08.2000 - Søren Hebsgaard

Mødedeltagere: Søren Hebsgaard og MAOV-gruppen

Formål: Identificere brugere, og høre Sørens ideer og forestillinger.

Vi har snakket om muligheder og begrænsninger ved projektet.

Eksterne grupper

Kan ikke lade sig gøre alligevel, da Oracle ikke tillader eksterne personer adgang til deres net.

WAP begrænsninger

- lille display
- elendig indtastninger
- simple opslag er det kanon til
- Indtaste en ordre - urealistisk

Efterlad ikke en usikker kunde:

Hvis handelen ikke kan gennemføres pga manglende oplysninger, efterlades en usikker kunde

- Der skal endnu et (dyrt) møde til.
- Konkurrenten kan springe til og stjæle kunden

Nødvendighed af systemet

- Hvor tit skal du ringe hjem når du er i marken
- Hvor mange simple action-points skal du hjem og lave
- Kan kunden føle sig usikker, når sælger forlader kunden pga. manglende info
- Hvor mange af de ting er simple

Virkelig forretningsmæssig værdi

- Undgå flere møder
- Kortere cyklus - billigere og nemmere.
- Bedre kundepleje

Udarbejd spørge-forarbejde - kontrol af Søren

Eventuelt spørgeskema - skal være kort.

Forskelligt hvor vigtige funktioner er:

- Nice to have 80% (eMail typisk)
- vs.
- Need to have 20%

Funtioner der endnu er teknologisk umulige:

Konsulenter får adgang til deres dokumenter - ikke terminaler idag, men sandsynligvis om et år.

Kalender-funktion: (typisk ønsket funktion)

Alle har sin egen kalender (papir/Palm), så det er andres kalender på WAP der er interessant (med booking)

Sidder ved møde og skal bruge en anden mand, kan han?

Lokations-bestemte tjenester - ikke realistisk p.t.

- Tjenesten kun tilbudt når den er interessant.
- Bestil en taxi med eet tryk, fordi fonen ved hvor den er.



Voice Over XML: Oplæsning af eks. eMails og talegenkendelse/verbale ordre - kanon i bil.

Afdelinger/navne som ville give mening

(find både folk der er meget interesseret, og folk som siger: "hvad skal jeg dog med det?")

Sælgere (ligger vandret før den første) - 80 pers.

Large Account - de tunge drenge

Jacob Broberg Jensen - Accountmannager - rolig nu type

Jesper Sachman - Salgschef - teknofreak

Peter Michael Jensen - Salgschef i dot.com

General Business

Thomas Christoph - Channel Account Maneger

Tonny Kolling Kristensen - Teknikmongol

Tina Lindholm - går efter bredt markede - større antal kunder

PSS

Michael Skårup Christensen - Nordisk behov - CRM afdeling mobilt - tænker bredt.

Jens Peder schou - gruppeleder

Supportere

Spørg Mogens

Bjørn Johannsen - Manager

Menig:

Anne Green - spørg eventuelt om andre - hvem er meget ude.

Remote DBA

Konsulenter - 250 pers.

Hvem-som-helst

Application consulting - oftest lange projekter

Technology consulting

Flere afdelinger

Dan kristensen Gruppeleder timerapportering

Hans Henrik Hoffmeyer

Operations (ikke væsentligeste) - 12 pers.

Kristian Bak (måske ikke tid)

Education

Karen Løkke Kristiansen

Marketing

Henrik Bustrup - alle kender ham

Egne spørgsmål:

Hvor ligger data, kan vi få SQL adgang...

- Tal med Asger - han viser også prototype/demoversion af MAOV.

Møder med brugere?

- Se ovenstående.

Test med WAP-fon?

- Brug emulator.

Portal to Go kursus - hvornår?

- Satser på uge 37 - internt kursus muligvis før.

Hvem er XML og JDev kyndig?



- Malte (spørgsmål og teknik) - brug hellere Asger, da han er tilknyttet projektet.

28.08.2000 - Asger Jensen

Mødedeltagere: Asger Jensen og MAOV-gruppen

Formål: At kunne komme igang med den tekniske del af projektet.

Vi har snakket om tekniske krav og muligheder, og Asger vil gerne sætte os i gang med Portal-to-Go, og hjælpe os med at lave et simpelt "Hello World"-lignende program, der smider sit output fra en DB, igennem Portal-to-Go til en WAP-telefon. Asger har selv kodet en del op imod Portal-to-Go, som vi kan få lov til at studere for at lære det.

Egne spørgsmål:

Kan vi få lov til at se på noget Portal-to-Go?

Vi har set klient-siden, og dens træ-struktur, på en lille tour, og hvordan outputet ser ud, både på WEB og WAP.

Kan vi se prototypen af det MAOV-lignende system der blev lavet for sjov?

Vi fik set et eMail-læse-system der kører over PtG. "Prototypen" lå ikke klar på serveren så den må vi se en anden gang. Den skal flyttes alligevel.

Hvilke databaser skal vi tilgå?

Vi får adgang til en DB som har nogle views med links til de andre databaser som indeholder de relevante oplysninger. Vi kan så læse det hele, men det er usandsynligt at vi får skrive-adgang. Vi får dog lov til at skrive oplysninger i vores egne tabeller i den lokale DB.

Kan vi få nogle servere til at køre den tunge del på, og kan vi få klient-delen af PtG?

Portal-to-Go ligger allerede på en server, med kørende udviklingsværktøjer. Vi har fået lov til at anvende den. Det vil sige at vi altså ikke selv får brug for vildt store hardwarekrævende maskiner. Klient-delen har vi allerede fået kopieret over. Det er desuden muligt at bruge JDK som vi er vant til, og som heller ikke er særligt krævende.

12.09.2000 - Asger Jensen, Søren Hebsgaard

Mødedeltagere: Asger Jensen, Søren Hebsgaard og MAOV-gruppen

Formål: At afgrænse funktionsforslagene til et par enkelte som er realistiske og brugbare.

Kalender for samtlige medarbejdere

Meget brugbar, men urealistisk teknisk og organisatorisk. Kalendersystemet er ikke et officielt produkt, vil ændre sig. At bruge det der er nu (strippe fra web) ville være et projekt for sig selv.

Telefonliste over medarbejdere

Simpel, der er adgang til alle data. Rent organisatorisk findes der ingen oplysninger om hvad hver enkelt kan.

Time-registrering

Igen meget brugbar, men ikke teknisk mulig = lukket land. Web-interface er java-applet.

Marketings-oplysninger

Ekstrem relevant, data er tilgængelige. Tilmelding skal ske pr. e-mail.



Adgang til sine e-mails

Kan godt lade sig gøre. Kun inbox er tilgængelig, ikke nogen andre folders. Det bliver muligt at svare på/sende e-mails, nogle standard tekster laves som hjælp.

Oplysninger om software-licenser

En stor ønsket funktion, der godt kan lade sig gøre. Snak med Betina Michelsen om db-link.

Prisliste (produkter og konsulent-ydelser)

En fed funktion, men ikke teknisk mulig. Viste sig alligevel at blive mulig efter møde med Christian Fabricius afholdt d. 17.10.2000.

Kontaktliste over kunder/partnere

Kan godt lade sig gøre. Snak med Rasmus Menchke om CDM-kundedb.

Oplysninger om aktuelle sager (support)

Dette er også muligt, men det er kun muligt at trække data ud om danske kunder.

Adgang til et udsnit af "Sales Online" systemet

Der er et stort behov for denne funktion, den har stor forretningsmæssig værdi. Men den er heller ikke teknisk og organisatorisk mulig. Man kunne muligvis strippe fra web, men det er et projekt i sig selv.

Har kunden betalt for tidligere køb

Ikke relevant for Oracle, kendte kunder, kreditgodkendes. Og desuden er det "kun" software der sælges.

Tildele action-points til kolleger

I relation til en kunde er det "Sales Online". Enkelt actions = send mail, cc en selv.

Endvidere blev der diskuteret en del om brugerinterface-design på mobileenheder. Hvordan skal man opbygge en applikation på sådanne devices. Man skal tænke meget over forskellige indgange til systemet.

Noget andet som Asger mener vil blive mere og mere anvendt er de såkaldte "intelligente agenter", der ligesom pusher data til brugeren. F.eks kan man få en agent til at sende trafikoplysninger til en på bestemte tidspunkter af døgnet.

22.09.2000 - Jørgen Karrebæk

Mødedeltagere: Jørgen Karrebæk og MAOV-gruppen

Formål: Statusmøde.

Vi gennemgik alle vores dokumenter systematisk for Jørgen og han kom så med kommentare til dem. Følgende kommentare fremkom:

Kravmodellen

Der var uklarhed om hvad PDOM'en viste, vi kom frem til at den er et udsnit af det MAOV gør brug af, dvs. et virtuelt view over det MAOV gør brug af i det eksisterende system.

I funktionsafgrænsningen "Telefonliste over medarbejdere" var det ikke klart nok beskrevet hvorfor det ville være umuligt at søge på faglige ressourcer. Af mock-ups'ne var det helt nemt at forstå hvordan der navigeres rundt på i MAOV-systemet. For at gøre navigeringen på WAP telefonen mere forståelig, blev det besluttet at der skulle udarbejdes et navigations-diagram.



Analysemodellen

ICE-modellen var for uoverskuelig, det blev besluttet at dele den op i 2. En ekstern og en intern.

Design

Vi kom frem til den arkitektur vi bygger systemet op over er en lag-delt arkitektur. Dette skal dokumenteres mere grundigt.

Andre ting der blev diskuteret

Det skal gøres klart fra hvilke databaser der læses og skrives. Vi har kun læserettigheder til de eksisterende interne Oracle databaser. Det er kun muligt at skrive til vores egen database, hvor diversn brugeroplysninger skal gemmes. Man kan endvidere sige at det at sende en e-mail er en form for at skrive til en database.

I design skal de enkelte databaser og tabeller, som MAOV benytter, beskrives. Samt egen tabeller. Og det skal være helt ned på attribut-niveau.

Der skal skrives om teknologivalget i indledningen til design, fordi dette blev allerede valgt da projektet startede, og kan ikke ændres. Dvs. forskellige teknologier skal ikke overvejes under design.

Program-koden skal ikke vedlægges som bilag, da dette ikke har nogen mening. Den kan istedet vedlægges på CD, og så henvise til den i rapporten. Det skal dog ikke forstås på den måde at der ikke må indgå noget kode i rapporten, hvis relevant for nogle designovervejelser/implementationsovervejelser så inddrag gerne lige præcis denne kode.

Forklar om Oracle's politik (globale IT-strategi), og hvorfor systemet ikke er officelt accepteret, og problemerne herved.

Skriv om den måde vi designer og implementerer på, det foregår via en såkaldt round-trip metode, som er helt anerkendt.

02.10.2000 - Diverse skolevejledere og andre projektgrupper

Mødedeltagere: Diverse skolevejledere, andre projektgrupper og MAOV-gruppen.

Formål: Præsentere vores projekter, og se hvor langt vi er nået.

Følgende 3 projekt-grupper var mødt op ud af de 5 der skulle være kommet:

1. Gruppe: MAOV

Thomas T. Pedersen
Martin Atke Bentsen

2. Gruppe: Axapta XML

Jakob Preysz Jørgensen
Johan Nis Bjørn
Peter G. Hansen

3. Gruppe: Cluster

Morten Mali

Vejledere:

Flemming Rix Jensen
Hanne Håkan
Svend Aage Filtenborg
Jørgen Karrebæk
Erik Buch Jakobsen



Vi præsenterede vores projekt ud fra:

- Hvad vi laver
- Hvilket firma
- Hvordan vores system hænger sammen
- Hvor langt vi er nået i tidsplanen
- Hvordan vi designer systemet

Det blev diskuteret:

- PDOM - ikke beskrivelse af de faktiske databaser.
- Databaserne skal beskrives!
- De var lidt bange for at vi kun havde programmeret uden at dokumentere, hvilket jo ikke er tilfældet.

Det var et udemærket møde, og sjovt at høre hvordan det gik med de andre grupper. Det var dog ikke fordi vi havde noget at hjælpe hinanden med. 2. Gruppe arbejder også med XML, men da det er så vidt et begreb, var det ikke muligt at hjælpe hinanden teknisk på det punkt. 3. Gruppe har intet tilfælles med det vi laver, da det var en teoretisk beskrivelse af "cluster-systemer" (flere computere der samarbejder om at løse problemer).

Sammenligning af hvor langt vi er nået i forhold til "2. gruppe". Vi er nogenlunde lige langt. De har dog skrevet mere end os, bl.a en del mere process-beskrivelse. Det er dog ikke grund til at vi skal bekymre os, da vi mistænker dem for at skrive om ting der ikke har relevans for vores projekt. Vi koncentrerer os om at udvikle et system, og dokumentere løsningen, uden at skrive for meget ævl om selvfølgeligheder. De har ligesom vi, fået implementeret en smule og fået hul igennem, så de ved at det kan lade sig gøre, men er langt fra færdige. De havde lidt problemer med at forklare deres design, men ellers gik det fint med dem. De har dog væsentligt flere stridigheder i gruppen en vi har...

...Efter mødet, snakkede vi lidt med vores egen vejleder (Jørgen Karrebæk). Han gav os stadig indtryk af at vi er på rette vej. Han kan jo følge med på vores hjemmeside, og se hvad vi får lavet. Dagen efter modtog vi en e-mail hvori han kommenterede et par detaljer.

05.10.2000 - Rasmus Mencke

Mødedeltagere: Rasmus Mencke og MAOV-gruppen

Formål: At få en overblik over CDM-database tabellerne.

For at kunne lave kontakt- og aktivitets-funktionen i MAOV, kræver det en hvis viden om de tabeller disse funktioner skal få sine data fra. Disse oplysninger ligger i Oracle's CDM-database. Efter lang tids analysering af disse tabeller, og uden tilfredsstillende resultat, kontaktede vi Rasmus, som skulle have en hel del viden om lige præcis denne database.

Vi præsenterede for Rasmus hvilken funktionalitet de to funktioner skal have, hvorefter han tegnede en model over de relevante tabeller og hvordan de hænger sammen. Dvs. hvilke attributter der linker de forskellige tabeller sammen.

Han viste os også nogle tilsvarende funktioner, som Oracle har på deres intranet. Så vi kunne få en smagsprøve på hvordan det ser ud på webben.



17.10.2000 - Christian Fabricius

Mødedeltagere: Christian Fabricius og MAOV-gruppen

Formål: Første test-snak.

Umiddelbart test-resultat:

Kan godt lide ideen med at den husker søgte kunder.

Søgning på medarbejdere er hurtig, men på kunder er den langsom (må godt optimeres !!!)

Har kunden aktive licenser

Prioriter rækkefølgen af kunder, så dem med licenser kommer først.

Skriv bynavn med når kunder listes, så man kan se hvor de kommer fra, og eventuelt en stjerne hvis de har licenser.

HUSK: Der er både aktive og terminerede licenser.

Der skrives for mange aktiviteter ud - sæt distinct på.

TAR - fejlmeddelelse på

Prioritet, dato for seneste opdatering, og hvem har bolden.

Prioriteret orden.

Prisliste mangler

- CF vil selv oprette en prisliste tabel og vedligeholde den, så den kan komme med i MAOV.

Action-points

- Ring til mig!!!

Lars Selsbæk - support-chef burde vide noget om TAR

lselsbae@dk.oracle.com

17.10.2000 - Bo Nielsen

Mødedeltagere: Bo Nielsen og MAOV-gruppen

Formål: Skaffe konkrete oplysninger om hvordan man læser fra TAR-databasen.

TAR-databasen indeholder data om aktuelle sager en kunde har med Oracle som venter på at blive afklaret. Det er dog kun de sager der er aktuelle som MAOV-systemet skal have fat i. Hver TAR har en status i form af en talværdi, og det var den vi skulle have afklaret hvordan den skulle tydes.

Bo Nielsen er konsulent hos Oracle og ved hvordan TAR-databasen skal tydes. Vi blev henvist til ham fra Lars W. "Selsbæk", som vi blev henvist til fra Christian Fabricius (møde 17.10.2000).

Tabellen tar_head indeholder oplysninger om de enkelte TAR'er, og har felterne TAR_STATUS og BUG_STATUS, som begge bare er numre. De enkelte numre er forklaret i tabellen tar_status, som vi ikke kendte noget til, før dette møde. Når man linker de to tabeller sammen kan man få en tekst-beskrivelse af status for de enkelte TAR'er.



Der er dog de problem at der er to felter der skal linkes imellem tabellerne, og der skal derfor laves en dobbelt join, eller rettere sagt to "outer-joins". Bo Nielsen sammensatte en SQL-forespørgsel der henter de relevante oplysninger, og det er den vi har brugt i implementeringen.

Vi lavede derudover en gennemgang af felterne, og en af overraskelserne var at kontakt-personen som stod i databasen var kontakt-personen hos kunden, og ikke hvilken oracle-medarbejder der er tilknyttet den TAR.

Mødet varede ca. 20 minutter, og var aftalt uformelt, så det kunne være afbrudt af en kunde-henvendelse, hvilket det dog ikke blev.

02.11.2000 - Asger Jensen

Mødedeltagere: Asger Jensen og MAOV-gruppen

Formål: Accepttest af MAOV-systemet af primær kontaktperson.

Asger synes det er et rigtigt godt system og yderst brugbart. De funktioner som prototypen manglede er blevet implementeret, og det er han meget glad for. Desuden ser han det som vigtigt for brugbarheden, at systemet husker den enkelte brugers seneste valgte kunder og medarbejdere, når det primært er en WAP-telefon systemet skal køre på.

Ham havde dog nogle kritikpunkter:

- menupunkter der fører brugeren gemmen systemet manglede diverse steder
- ville gerne have at man kan se hvilke personer der er tilmeldt en aktivitet
- kontaktpersoner og aktiviteter måtte gerne hænge mere sammen

Men udover disse kritikpunkter, så kunne han godt godkende systemet.

Vi diskuterede desuden om hvad et db-link er.



8.14 Bruger-interviews

Alle afholdte interviews er at finde i det følgende. De er sorteret i kronologisk rækkefølge.

28.08.2000 - Dan Kristensen

Afdeling: Consulting
Stilling: Consulting Manager
Navn: Dan Kristensen

Hvor meget tid (ca. %) bruger du ude af huset:
60% (alm. konsulenter typisk 70%)

Umiddelbare ideer til funktioner:

Aftale møder på andres vegne. Skal nu ringe på mobil-telefon - ikke altid til at få fat i.
Telefon-liste - erstatning for udprintet papir-version.
Time-registrering. Foregår nu på et Oracle-web-system. Stort irritations-moment - proppet med langsomme applets, kører dårligt, dårligt til at håndtere flere projektet. Time-registrering skal være indrapporteret hver mandag.

Primær arbejdsfunktion:

Dan er gruppeleder for nogen konsulenter, og det er ham der får det første konsulent-møde med kunden, forhandler kontrakten og tildeler så opgaverne til sine konsulenter.

Hvordan forløber en typisk sagsbehandling i grove træk:

Hvis det ikke er en simpel sag, tager Dan ud til kunden for at høre om sagen. Derefter tager han hjem på kontoret igen og udarbejder en kontrakt udfra et standard aftale-brev, og tilføjer en opgavebeskrivelse. Ved simple sager, kan brevet udformes allerede inden første møde.

Hvis der skal bruges en specialist til at løse en opgave, som ikke er under Dan, kan det være besværligt at finde frem til den rette, og få ham tilknyttet projektet. Han har overblik over hvad hans egne konsulenter kan, men ikke hvad de andre kan. Når han endelig finder en, er han/hun måske ikke til at få fat i.

Hvordan sætter du dig ind i en sag, når du skal til at starte/genoptage en:

Er så major grundigt forberedt at det ikke er noget problem at holde styr på sine sager.

Hvilke oplysninger skal du ringe hjem om:

Ringer aldrig hjem til kontoret, da han altid er meget grundigt forberedt. Hvis der alligevel er spørgsmål om produkter, problemer og løsninger, kan han finde svar på Meta-link på internettet. Der er som regel adgang til nettet med bærbar og modem, når konsulenterne arbejder hos kunden - men Meta-link kan ikke bruges til møder!

Hvilke simple oplysninger noteres, som skal registreres hjemme:
Time-registrering.

Efterlades kunder med uafklarede spørgsmål. Hvilke spørgsmål:

Åbne punkter, skal bare ordnes. Kunden skal være afklaret om hvilke



spørgsmål det drejer sig om, og være forvisset om at Dan melder tilbage med svarene hurtigst muligt.

Er der noget der irriterer dig når du er ude, pga. manglende oplysninger/muligheder. Hvilke:
Igen, booking af møder med andre konsulenter over mobil-telefon.

29.08.2000 - Henrik Bustrup

Afdeling: Marketing
Stilling: Solution Marketing Manager
Navn: Henrik Bustrup

Hvor meget tid (ca. %) bruger du ude af huset:
5% (ikke hos kunder)

Umiddelbare ideer til funktioner:
Marketing er egentligt ikke ude af huset, og har derfor ikke noget reelt behov for mobil adgang til oplysninger. De har dog nogle bud på hvad sælgerne kunne have brug for af marketingsoplysninger.

- Hvilke seminarer kunden er tilmeldt
- Hvilke nyhedsbreve får kunden (mailinglister).
- Hvem er tilmeldt et givent seminar/mail-liste
- Adgang til sine e-mails (som Asger udvikler)

Kunde spørger:
Har jeg fået det nyhedsbrev?
Er jeg tilmeldt det seminar?

Sælger bruger "oracle sales online" - alt hvad der kan trækkes ud af det vil være af værdi ude af huset.
Marketing får tilsvarende program, men har ikke noget behov uden for huset.

Kender kunden til et arrangement? Er kunden tilmeldt? Kan sælgeren referere til det under mødet?

Henrik foreslår følgende relevante personer vi kan tale med:
- Michael Odfeldt (sælger)
- Heidi Linderup (sælger)

Hvordan forløber en typisk sagsbehandling i grove træk:
- ingen sagsgang!

Hvordan sætter du dig ind i en sag, når du skal til at starte/genoptage en:
- ingen sagsgang!

Hvilke oplysninger skal du ringe hjem om:
- ikke ude!

Hvilke simple oplysninger noteres, som skal registreres hjemme:
- ikke ude!

Efterlades kunder med uafklarede spørgsmål. Hvilke spørgsmål:
- ikke ude!



Er der noget der irriterer dig når du er ude, pga. manglende oplysninger/muligheder. Hvilke:
- ikke ude!

30.08.2000 - Kristian Bak

Afdeling: Finance
Stilling: Finance Director
Navn: Kristian Bak
Formål: Høre om Oracles globale IT-strategi, og behov for MAOV i Finance o.lign.
Ekstra deltager: Claus Torp Hansen (IT-ansvarlig)

Oracle har en strategi der går ud på at alt software skal udvikles så det kan bruges globalt. De har samlet alt udvikling i USA, og det betyder at de enkelte lande ikke må udvikle deres egne systemer. Umiddelbart virker det som en meget streng og uflexibel strategi, men den har flere betydelige fordele. Strategien blev indført ved årsskiftet 1998/99, og har i '99 medført en besparelse på 1 mia. dollars.

Når de enkelte lande udvikler deres egne systemer, kan de sjældent bruge hinandens, og der er en masse problemer med at få dem til at kommunikere sammen. Der er heller ikke styr på hvilke systemer de enkelte lande har udviklet. Det kan betyde at flere lande kommer til at udvikle de samme systemer, og derved bliver de samme udviklingsomkostninger betalt flere gange.

Når alle systemer bliver udviklet centralt, betales udviklingsomkostningerne kun een gang, og når systemerne implementeres i de forskellige lande, kan de med sikkerhed kommunikere sammen. Desuden vil alle afkroge af Oracle's afdelinger hurtigt kunne udnytte opdateringer af systemerne, og man undgår at eet land sidder med et forældet system der ikke har været opdateret i lang tid.

Køb af hardware

Oracle Danmark begrænser deres omkostninger til hardware, til kun at bevilge når noget er strengt nødvendigt. Det er ikke sandsynligt at Oracle Danmarks medarbejdere får den nødvendige hardware, til at bruge vores MAOV-system, og vi er derfor blevet advaret imod at give folk forventninger om at kunne få gavn af det. Der er p.t. kun ca. 10-20% der har WAP-telefon og de er primært placeret i Platform-afdelingen. Det er meget få sælgere der har WAP. De få der har Palm har oftest købt dem for deres egne penge, da Oracle mener at de nuværende Palm-Pilots ikke opfylder de nødvendige tekniske minimumskrav.

Sikkerhed

Oracle har strenge sikkerhedskrav, og det vil derfor ikke være muligt at lave vores system så det kan bruges fra mobile enheder. De tillader ganske enkelt ikke adgang udefra. Hvis de endelig skulle implementere et sådanne system, skal Oracles globale strategi tillade dette.

Repræsenterer alt back-office.

Finance er aldrig ude, max een gang om måneden.



Backoffice, HR, økonomi har reelt intet behov.
Indrette på fast arbejdsplads med faste program

Betragt det derfor som en øvelse, begræns det.
Vi skal derfor kun betragte projektet som en øvelse, og koncentrere os om det der giver gode karakterer.

Passe på medarbejderes forventninger, vi skal ikke rulle frem med et eller andet som alligevel ikke kan tages i brug.

Yderligere kontakt-personer:

Christian Drewer (intern service) - uddeler mobiltelefoner og ved hvor mange der har WAP. Vi spurgte ham, han svarede, at de har 45 WAP-telefoner p.t. fordelt på alle afdelinger.

07.09.2000 - Christian Fabricius

Afdeling: Sale
Stilling: Sales Consultant
Navn: Christian Fabricius

Han spørg:

Hvor langt er vi???
Vil det blive vedligeholdt???

Har snakket med andre sælgere om hvad de kunne have brug for...

Hvor meget tid (ca. %) bruger du ude af huset: meget!

Umiddelbare ideer til funktioner:

Funktioner i WAP-systemet skal være ting man kan klare når man er ude af huset. Der vil altid være mange opgaver der kræver at man er face to face med kunden / kollega'en, som ikke vil kunne erstattes af MAOV. Der er store dele af arbejdet der udføres gennem dialog.

Interessante områder for MAOV kan være oplysninger om software-licenser ude hos kunderne. Oracle sælger licenser, og support, konsulenter og uddannelse kommer så på som tillæg.

Eks. ønsket situation: Kunden spørger: "hvad har jeg egentligt?"
Sælgeren trykker kundens id e.lign og kan bladere gennem kundens licenser m.m.

En sælger har typisk 5-15 navngivne kunder af gangen, og det ville være rart at kunne finde oplysninger om dem, uden at skulle søge alle kunder igennem. Man kunne eksempelvis have en 4-bogstavs forkortelse for de navngivne kunder. Det ville være elegant!

Der findes allerede et system til at finde licens-oplysninger på intranettet, men der er ikke nogen ordentlig front-end til, og det begrænser også meget at det kun ligger på intranettet.

Flere virksomheder er delt ud i afdelinger med hver deres licenser (ex. Novo Nordisk).

Der er et sted på intranettet med licensoplysninger.
Der er mange oplysninger der ikke er særlig relevante. (15 kolonner pr produkt)



De Vigtige oplysninger er:

- hvilke produkt
- hvor mange licenser
- hvilken platform kører det på

Det kan være kompliceret hvilke afdelinger der har hvilke licenser.

Den mest direkte måde at finde licenserne på er papir-kontrakterne - direkte adgang!!!

Yderligere kontakter:

Betina Michelsen - link til eksisterende licens-frontend

Prisliste:

Tilgængelige på intranet

Selve den opdaterede liste ligger i MS-regneark (tihi)

Birgit Kuhl - ved mere om prislister

Der er reelt kun 3 pristyper for hvert produkt, så det er ikke så kompliceret.

Når der sælges et produkt er der en liste følgeprodukter for hver - nogle følgeprodukter er krævet for at systemet kører, mens andre kun optionelle, som frit kan vælges til eller fra.

Grundlæggende 2 kategorier af produkter:

- CORE (DB/JDev)
- Application (finans/brancesystemer)
- vigtigst med CORE da det er rimelig fast varesortiment, og der sælges meget af det.

Prisliste ligger idag i forskellige udgaver - der må ideelt slet ikke være en papir-udgave, da der så er mulighed for at den er forældet!!!

Konsulent-prisliste:

- hvad koster en konsulent med givne kvalifikationer. Det kan være svært at gennemskue idag.

Telefonnumre:

Styres af marketing. Det er ikke altid nemt at finde. Det kunne være rart at kunne søge i samtlige numre - også alle kunders. På den måde er man sikker på altid at kunne finde et specifikt nr.

e-mail: (ønsket af chefen)

Hvis man venter en eMail der har indflydelse på et kundemøde

Visheden om at man er opdateret med sin indbox

VIGTIGT er, om der er kommet noget - ikke at kunne skrive eMails (umuligt alligevel med WAP!)

CF har været udsendt til Berlin - Havde bærbar med, men det er totalt dyrt at koble på med modem fra hotellet. Det ville lettere kunne forsvarliggøres at bruge WAP end MODEM, da de begrænsede display ville begrænse brugen til det mest nødvendige... Oracle har nogle opkaldscentre verden over, men det virkede ikke særlig godt. Så er det bedre med WAP, da det er mere centralt, og derfor med større sikkerhed opdateret.

TAR - kun supporterne!

Straks når kunder ringer, oprettes sag som opdateres siden opdateres.

Er der vigtige sager der står og venter (prioriteter).

Er det kunden der har bolden eller Oracle, (hvem venter på hvem)

Der kan være det problem at når en sælger kommer ud til en kunde og tror han skal sælge noget, så risikerer han at kunden har problemer



med nogen produkter, og så skal han kun høre brok. Det ville være smartere hvis han lige kunne slå op om den slags problemer, og så starte mødet med at sige: "Jeg ved der er problemer med de to sager, hvordan går det med dem?" Så er man på det rene, og kan komme videre med at sælge noget!

Marketing-info: God ting med marketing seminarer.
Hvilke seminarer kommer de næste 3 måneder.
Kun kort oversigt - ikke smart med beskrivelse af seminar på WAP!
Kan ikke sige til kunde - prøv at se her på WAP'en - det holder ikke.
Hans eget udtryk: Der skal kun være hjørner af informationerne på WAP.

Der er typisk 10-15 nøgle-kunder (navngivne) pr salgsansvarlig. De er knyttet til en primær sælger.

På tværs af de salgsansvarlige er der nogen mere tekniske orienterede personer. De er tilknyttet flere salgsansvarlige, og har derfor typisk flere kunder tilknyttet end dem.

Yderligere kontakt-personer
- Lars Selsbek - salgschef
- Marketing - styrer registrering af telefonnumre

Der er ofte en halv time op til et kundemøde der kan bruges til at holde sig up-to-date med en WAP-telefon, og ligeså efter mødet er der tid til at WAP'e.

Sælgere har aldrig bærbare med til kundemøder, han har ihvertfald aldrig oplevet det.

07.09.2000 - Michael Skaarup Christiansen

Afdeling: Sales
Stilling: Sales Consultant, Nordic
Navn: Michael Skaarup Christiansen

Hvor meget tid (ca. %) bruger du ude af huset: 60% (dækker hele norden)

Michael Skaarup sidder i en presales/funktion, der vejleder kunden teknisk om hvordan Oracle-programmerne fungerer.

Områderne er delt...

Hovedområde:

- DataBase
- Udviklings-software
- E-Business suite (ligesom SAP / Axapta) - MSC's område!
 - Buy-side - indkøb / logistik
 - In-side - HR / finans / økonomi
 - Sell-side - CRM (Customer Realations Manegement) - Det der sælger!
- Marketing
- Sales >>VIGTIGST<< ("sales online" skal gøres mobilt)
- Service
- e-Business
- Call-center



Sales Online - mest efterspurgt del af kunder - brugt i Oracle af:
- account managere
- pre-sales
- ledelse

Hvad man skal se på WAP: (forecast-salgsværktøj)
salg/muligheder/kontakter

Systemet findes til Palm, men kunderne spørger: hva' med WAP?
Hvis bare vi ku' vise simpelt: Log in - se kunder - opportunities
...det ville få direkte impact på salget af E-business-systemet.

Den største konkurrent har kun et log-op til Finland - Det ville være røvblæret at kunne vise en fungerende version, der kører i DK og hvor man eksempelvis kan slå kunden selv op!!!

ØNSKE: Kundeliste (kontakt/opportunities) - muligvis starte ud fra opportunities.

FORMÅL: Vis til kunder og til brug for account-managere

På "sales online" kan MSC selv kun se kunder der er interesserede i CRM. Han får dem tildelt af andre sælgere der mener at han skal snakke med kunden om netop det!

Hvis vi har for meget tid så gå videre til service-afdelingen!

"sales online" på <http://sales.us.oracle.com> - evt. hvordan på WAP
- opportunities - hvilke muligheder er der for salg
Se liste med kundenavne
klik for at se detaljerede oplysninger
- contact - hvem man har fat i hos kunden - der kan være flere personer/afdelinger
- customer - Hvilke kunder er tilknyttet denne sælger.
- ToDo - mindre vigtigt - reelt action-points. Der er få der bruger WEB-versionen

Yderligere kontakter:

Henrik Elkær - Ved alt om "sales online".

07.09.2000 - Peter Michael Jensen

Afdeling: Sales
Stilling: Sales Director
Navn: Peter Michael Jensen

Peter Michael Jensen bruger meget sin Palm, som han synkronisere hver dag med Oracle nettet. Han mener ikke det er nødvendigt på nuværende tidspunkt at kunne gå online med en WAP-telefon. Det er nok at have oplysningerne i sin Palm, som jo synkroniseres hver dag! Som Peter udlagde det "Gør ikke tingene mere kompliceret end det er! - Keep it simple small stupid". Eksempelvis så scanner Peter alle de visit-kort han får af kunder og uploader dem til sin Palm.

Peter kunne dog godt se at der kunne være oplysninger, der ville være rigtigt gode at kunne få fat på online. Af funktioner og oplysninger nævnte han:

Kontaktoplysninger på kunder, kollegaer og partnere. Det ville være meget anvendeligt og praktisk. Helt sikkert meget vigtig funktion.



Prisliste, bør kunne downloades på Palm, ikke så vigtigt på WAP.

Adgang til kalender-systemet, hvilket jo er en meget banal ting man gør hver dag! Er jeg booket? - reserver tid!

Hvilke maillister/kurser/seminare er kunden tilmeldt. Eller hvilke seminarer/kurser har kunden været på inden for de sidste tre måneder.

Hvis en kunde ikke har betalt sidste gang, vil du ikke sælge til dem igen. Det kontrolleres sjældent i online-butikker.

Adgang til sin e-mail inbox, kun kunne læse dem.

Action-points kunne være rare at oprette på stedet, f.eks det undersøger en bestemt person og ringer igen næste uge. Denne person skulle så blive informeret om dette, ved at man skal kunne tildele actions til hinanden (ToDo's). Men dette punkt er mere relevant for en Palm, dvs. når man kommer hjem synkroniseres Palm'en med Oracle nettet. Peter mener at det bør ske som koordineret del af Business-systemet.

Mange af disse funktioner ligger som standard i CRM-systemet, det er bare ikke implementeret her i Oracle Danmark. Information om dette kan fås ved at læse Oracle's CRM litteratur.

Dvs. Peters syn på mobileenheder er ikke at være online med dem, bare man kan synkronisere dem med hovedkvarteret et par gange om dagen, så er det rigeligt. F.eks da Peter tidligere var meget på farten (afrika/amerika/østen), downloadede han alle sine e-mails til sin laptop inden han tog i lufthavnen og læste/svarede dem i flyet og sendte dem så om aftenen når han igen var på nettet. I USA er dette meget brugt, for hvad skal man ellers lave i flyet, de er bare et skridt videre. Derover kan man gå online i flyet. Her er Europa generelt bagud, for dette kan ikke lade sig gøre endnu.

Hans pointe er: "Nice-to-have alting online - smartere at kunne downloade til Palm".

Vigtigt: Hvad er bedst til en given teknologi. Man kan operere med flere niveauer af mobillitet, Laptop, Palm, WAP. Bruge virksomhedens midler bedst. Han sagde også, at det at være kritisk overfor hvad der egner sig til de enkelte niveauer og kunne sortere i det. Det vil en censor lægge mærke til: "De har ikke taget alt med bare for teknologiens skyld!".

Peter har en fast kundeportefølge på ca. 15 kunder ad gangen.



8.15 Litteraturliste

Følgende litteratur er anvendt som grundlag for dette projekt. Store dele af projektet er dog udarbejdet ud fra egne erfaringer, og fra undervisningen gennem 4 semestres datamatikeruddannelse på Lyngby Uddannelses Center. Værkerne er nævnt i tilfældig rækkefølge.

1. James Rumbaugh m.fl.
Object-Oriented Modeling and Design
Prentice Hall 1991
ISBN: 0-13-630054-5
2. Lars Mathiassen m.fl.
Objekt Orienteret Analyse og Design
Marko 1997
ISBN: 87-7751-123-9
3. Andreas Munk-Madsen
Strategisk Projektledelse
Marko 1996
ISBN: 87-7751-115-8
4. Ivar Jacobson m.fl.
Object-Oriented Software Engineering
Addison Wesley 1998
ISBN: 0-201-54435-0
5. Frank Buschmann m.fl.
A System of Patterns
John Wiley & Son Ltd
ISBN: 0-471-95869-7
6. Gitte Henriksen & Michael Specht
Introduktion til Oracle programmering
Klim 1998
ISBN: 87-7724-756-6



8.16 Greetings

Vi hilser og takker følgende mennesker for at være der under udviklingen af projektet.

8.16.1 Ansatte hos Oracle Danmark

- **Asger Jensen** - Manden der har givet os en suveræn vejledning vedrørende PtG og et væld af andre emner.
- **Søren Hebsgaard** - For hele tiden at give indtryk af at alt kan lade sig gøre.
- **Mogens Nørgaard** - Som tillod os at arbejde her hos Oracle, og kort tid efter sagde op.
- **Christian Fabricius** - Superbrugeren som var til stor hjælp under tilpasningen af systemet til brugerne.
- **Rasmus Mencke** - Som forklarede CDM-databasen for os, og derefter rejste til USA.
- **Bo Nielsen** - Som hjalp os med at hive de korrekte data ud af TAR-databasen.
- **De interviewede brugere** - Der gav input til hvad systemet skulle kunne.
- **Receptionister** - Der altid var flinke til at hilse om morgenen.
- **Kantine-folkene** - Som lavede sund, ernæringsrigtig og afvekslende frokost til os hver dag.

8.16.2 Ansatte på Lyngby Uddannelse Center

- **Jørgen Karrebæk** - Vores flinke vejleder på skolen der har bakket os op gennem hele forløbet, og sagt at det ser da meget fornuftigt ud.
- **Hanne Håkon** - Som forsøgte at holde styr på hvad de forskellige grupper lavede.

8.16.3 Andre studerende der lavede hovedopgave hos Oracle.

- **Adam (Tsiah)** - Var den vilde spiller fra OCML-gruppen, som sørgede for sammenholdet ved at arrangere computer-spille-party hver fredag aften.
- **Ask** - Den "kloge" og eftertænksomme mand i OCML-gruppen, som også var med på computerspil om aftenen.
- **Tobi** - Tobias fra OCML-gruppen som rodede med en masse programmer, og luskede rundt i spil hver fredag aften
- **Yinghui** - OCML-gruppens engelsktalende hardcore-programmør, som programmerede næsten hele deres produkt, og sad og sov fra tid til anden.
- **KISS-gruppen** - Som også arbejdede med WAP, men ikke gennem PtG. De var ikke med på computer-spille-party fredag aften.
- **Allan og Lotte** - Der forsvandt midtvejs i projekt-forløbet. De havde problemer med projekt-grundlaget.