TMG Webbaseret ressourceallokeringssystem til projektplanlægning

Thomas Bergstedt

Kongens Lyngby 2007
IMM-B.Eng-2007-69

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

The title of my examination is ‘TMG Webbased resource allocation system for project planning’. The final result of this 10 weeks examination would be suitable for the department “GIS & IT”, in which I am working at Atkins, because I found the needs for a new webbased resource allocation system for work planning in my opening phase through interviews with employees in that department

The prototype for the planning system after the 10 weeks would consist of 3 parts.

- Part 1. A database on a SQL server, that are able to contain the needed data for the working planning. It should consist of data for employees, cases and projects and relation between those. The data will consist of elements from the 2 existing systems.
- Part 2. A web service, that consists of the needed routines/algorithms for the working planning etc.

1. Routines for creations of projects/cases and employees.
2. Routines for assigning/deleting/updating projects/cases on an employees working plan.
3. Routine for showing the resource allocation (work plan) for an employee.
4. Routine for showing/calculating the workload and outsourcing for each employee.

- Part 3. A very simple web application are made, which uses some of the implemented web service methods. The web application is only intended to show the use of the implemented web service and database

Resume

Titlen på mit eksamensprojekt er ”TMG Webbaseret ressourceallokeringssystem til projektplanlægning”.
Det endelige resultat af dette 10 ugers eksamensprojekt vil være dækkende for den afdeling ”GIS & IT”, jeg arbejder i hos Atkins, da jeg gennem interviews med medarbejdere fra denne afdeling har fundet behovene til et nyt webbaseret ressourceallokeringssystem til arbejdsplanlægning i min indledende fase.
Prototypen på planlægningssystemet efter de 10 uger vil bestå af 3 dele.

- Del 1. En database på en SQL server, som indeholder muligheden for at opbevare alt den nødvendige data for arbejdsplanlægning. Herunder data for medarbejdere, sager/projekter og relationer mellem disse. Data vil bl.a. bestå af elementer fra de 2 nuværende systemer.
- Del 2. En web service, der har de nødvendige algoritmer/rutiner til brug for arbejdsplanlægning bl.a.
    1. Rutiner til oprettelse af projekter/sager og medarbejdere.
    2. Rutiner til Tildele/Slette/Opdatere projekter/sager på en medarbejders arbejdsplan.
    3. Rutine til visning af ressourceallokering(Arbejdsplan) for medarbejdere.
    4. Rutine til visning/beregning af arbejdsbelastning og uddebitering for den enkelte medarbejder
- Del 3. En meget simpel webapplikation laves, som gør brug af nogle af de implementerede web service metoder. Webapplikationen er i dette projekt kun beregnet til testvisning af brugen af den opbyggede web service og database.

Forord

Dette projekt er mit afgangsprojekt fra IMM DTU, før jeg har opfyldt kravene for en B. Eng grad indenfor retningen IT. Projektet er blevet udført i samarbejde med ingeniør rådgivnings firmaet Atkins Danmark.

Dette projekt beskriver udviklingen af et webbaseret planlægningssystem med en SQL Server 2005 database, en web service. Desuden udvikles en simpel test webapplikation, som benytter nogle af web servicens implementerede metoder.

Projektet består af en rapport og en CD med kildekoden for den opbygget Web Service og Webapplikation, samt et script til oprettelse af Databasen.

Lyngby, Oktober 2007

Thomas Bergstedt

Anerkendelse

I udarbejdelsen af dette projekt har jeg haft lejlighed til at få råd og vejledning fra en del personer, men jeg vil især gerne takke følgende.

- **Jesper Skovdal Christiansen** – min virksomhedsvejleder, som gennem projektforløbet kom med gode råd og vejledning til projektet.
- **Morten Bork** – Som gav mig gode råd og vejledning med UP udviklings forløbet, og hvordan de enkelte UML diagrammer skulle laves.
- **Stig Høgh** – min DTU vejleder, som gav mig gode råd undervejs i forløbet men hensyn til strukturen af rapporten, og hvad jeg skulle fokusere på i projektet.
- **Ole Bergstedt –** Som læste og kommenterede min rapport, og kom med gode råd undervejs.

Indholdsfortegnelse

Kapitel 1

Introduktion

1. Introduktion

1.1 Baggrund for projektet

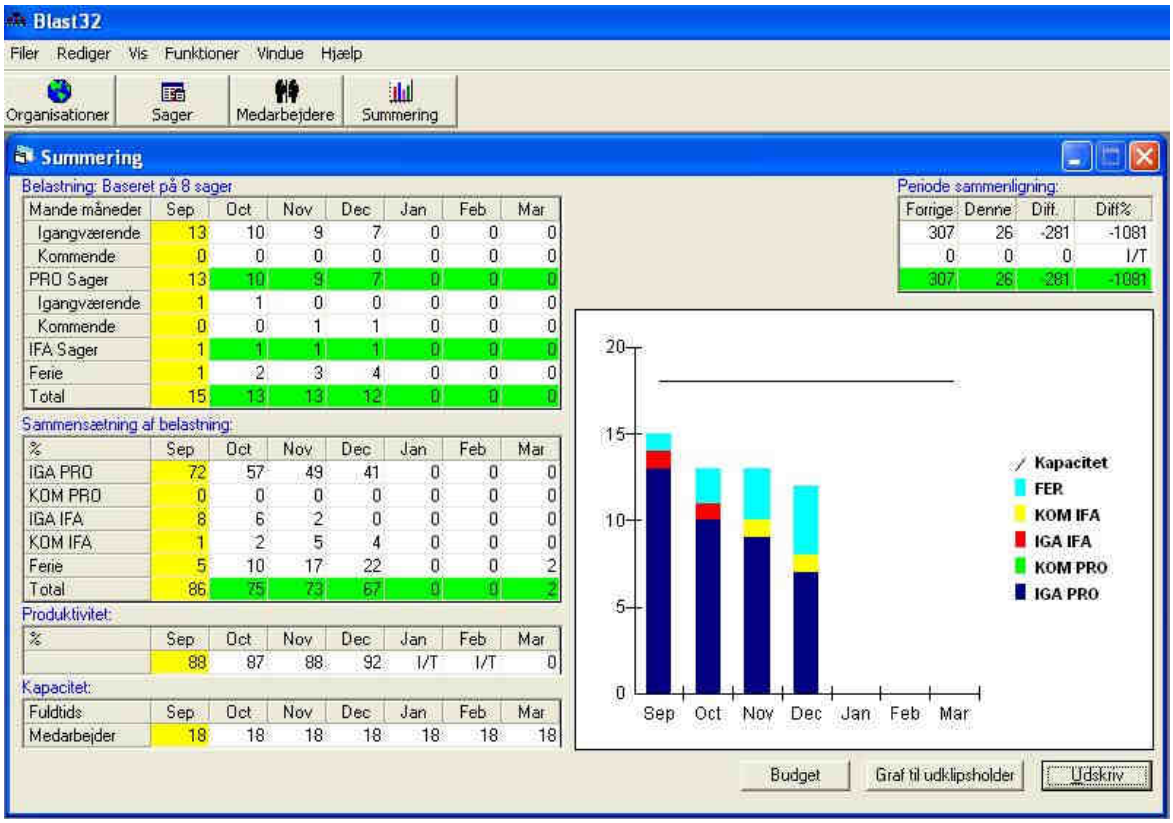
Sektoren GIS og Miljø hos Atkins Danmark er en stor sektor med mange medarbejdere. Sektoren består af en række afdelinger.

- GIS & IT
- Miljø
- Landmåling

I hver afdeling arbejdes der med mange projekter. Til planlægning af medarbejder ressourcer til nye og nuværende projekter har der længe manglet et godt redskab, som kan vise, hvad den enkelte medarbejder skal arbejde på i perioder, og hvad dennes arbejdsbelastning er med hensyn til månedens arbejdstimer og uddebiteringsgrad.

Til denne arbejdsplanlægning benyttes på nuværende tidspunkt 2 systemer.

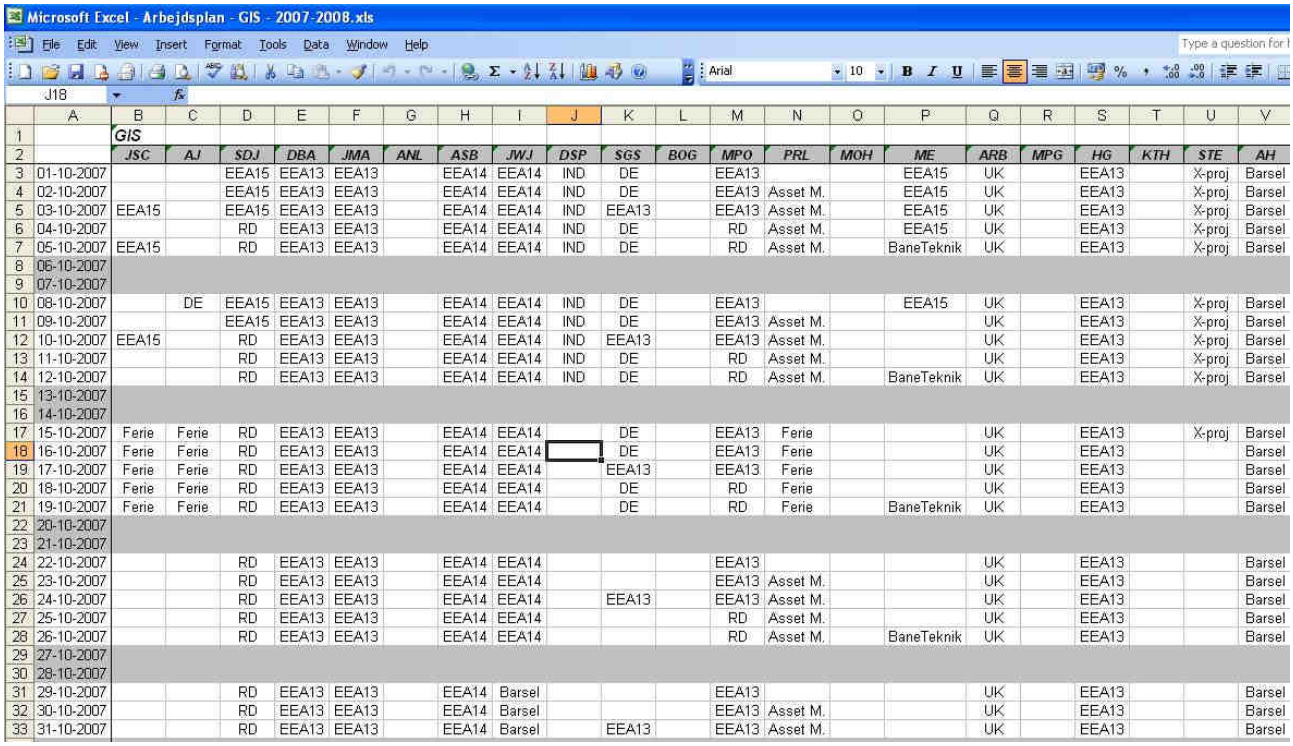
- Blast 32
- Excel ark med arbejdsplan og medarbejderbelastning.



Figur 1 Blast 32

Blast 32 er et system, som Atkins har haft i nogen tid, men som kun benyttes af afdelingslederne og cheferne. Dette system bruges primært til at lave nogle summeringer over medarbejderkapaciteten

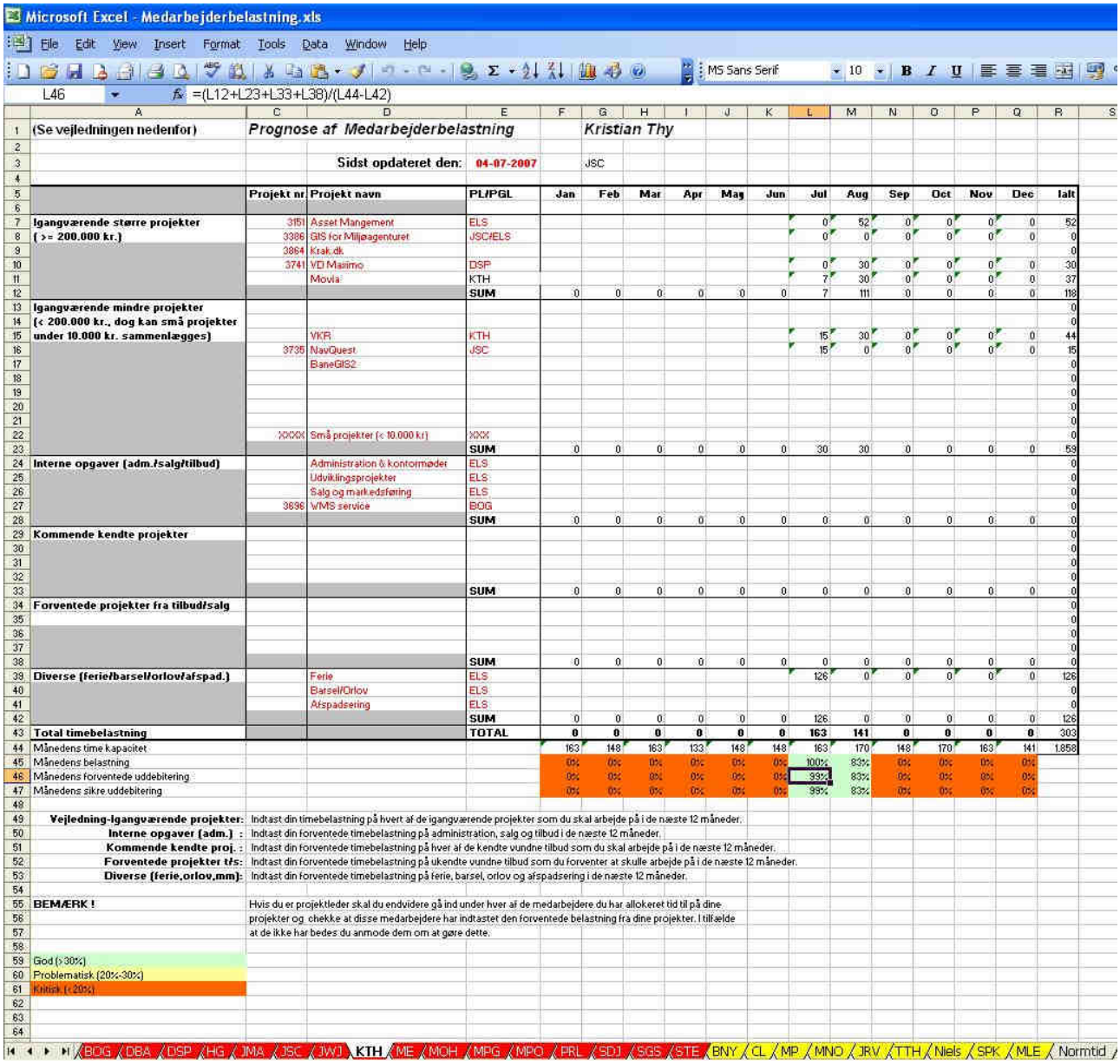
for måneder frem i tiden, så afdelingslederne kan få et overblik over, om der skal ansættes flere eller det modsatte.
Ulempen ved dette system er, at det ikke er så anvendeligt til en arbejdsplanlægning, og at systemet er svært at forstå med hensyn til, hvordan det laver summeringer over udtræk for økonomisystemet SAP.


Microsoft Excel - Arbejdsplan - GIS - 2007-2008.xls

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1		GIS																				
2		JSC	AJ	SDJ	DBA	JMA	ANL	ASB	JWJ	DSP	SGS	BOG	MPO	PRL	MOH	ME	ARB	MPG	HG	KTH	STE	AH
3	01-10-2007			EEA15	EEA13	EEA13		EEA14	EEA14	IND	DE		EEA13			EEA15	UK		EEA13		X-proj	Barsel
4	02-10-2007			EEA15	EEA13	EEA13		EEA14	EEA14	IND	DE		EEA13	Asset M.		EEA15	UK		EEA13		X-proj	Barsel
5	03-10-2007	EEA15		EEA15	EEA13	EEA13		EEA14	EEA14	IND	EEA13		EEA13	Asset M.		EEA15	UK		EEA13		X-proj	Barsel
6	04-10-2007			RD	EEA13	EEA13		EEA14	EEA14	IND	DE		RD	Asset M.		EEA15	UK		EEA13		X-proj	Barsel
7	05-10-2007	EEA15		RD	EEA13	EEA13		EEA14	EEA14	IND	DE		RD	Asset M.		BaneTeknik	UK		EEA13		X-proj	Barsel
8	06-10-2007																					
9	07-10-2007																					
10	08-10-2007		DE	EEA15	EEA13	EEA13		EEA14	EEA14	IND	DE		EEA13			EEA15	UK		EEA13		X-proj	Barsel
11	09-10-2007			EEA15	EEA13	EEA13		EEA14	EEA14	IND	DE		EEA13	Asset M.			UK		EEA13		X-proj	Barsel
12	10-10-2007	EEA15		RD	EEA13	EEA13		EEA14	EEA14	IND	EEA13		EEA13	Asset M.			UK		EEA13		X-proj	Barsel
13	11-10-2007			RD	EEA13	EEA13		EEA14	EEA14	IND	DE		RD	Asset M.			UK		EEA13		X-proj	Barsel
14	12-10-2007			RD	EEA13	EEA13		EEA14	EEA14	IND	DE		RD	Asset M.		BaneTeknik	UK		EEA13		X-proj	Barsel
15	13-10-2007																					
16	14-10-2007																					
17	15-10-2007	Ferie	Ferie	RD	EEA13	EEA13		EEA14	EEA14		DE		EEA13	Ferie			UK		EEA13		X-proj	Barsel
18	16-10-2007	Ferie	Ferie	RD	EEA13	EEA13		EEA14	EEA14		DE		EEA13	Ferie			UK		EEA13			Barsel
19	17-10-2007	Ferie	Ferie	RD	EEA13	EEA13		EEA14	EEA14		EEA13		EEA13	Ferie			UK		EEA13			Barsel
20	18-10-2007	Ferie	Ferie	RD	EEA13	EEA13		EEA14	EEA14		DE		RD	Ferie			UK		EEA13			Barsel
21	19-10-2007	Ferie	Ferie	RD	EEA13	EEA13		EEA14	EEA14		DE		RD	Ferie		BaneTeknik	UK		EEA13			Barsel
22	20-10-2007																					
23	21-10-2007																					
24	22-10-2007			RD	EEA13	EEA13		EEA14	EEA14				EEA13				UK		EEA13			Barsel
25	23-10-2007			RD	EEA13	EEA13		EEA14	EEA14				EEA13	Asset M.			UK		EEA13			Barsel
26	24-10-2007			RD	EEA13	EEA13		EEA14	EEA14		EEA13		EEA13	Asset M.			UK		EEA13			Barsel
27	25-10-2007			RD	EEA13	EEA13		EEA14	EEA14				RD	Asset M.			UK		EEA13			Barsel
28	26-10-2007			RD	EEA13	EEA13		EEA14	EEA14				RD	Asset M.		BaneTeknik	UK		EEA13			Barsel
29	27-10-2007																					
30	28-10-2007																					
31	29-10-2007			RD	EEA13	EEA13		EEA14	Barsel				EEA13				UK		EEA13			Barsel
32	30-10-2007			RD	EEA13	EEA13		EEA14	Barsel				EEA13	Asset M.			UK		EEA13			Barsel
33	31-10-2007			RD	EEA13	EEA13		EEA14	Barsel		EEA13		EEA13	Asset M.			UK		EEA13			Barsel

Figur 2 Excel ark for arbejdsplan

Det andet system består af 2 excel ark, som hver afdeling har sit eksemplar af. Det ene bruges til en arbejdsplan for afdelingen, hvor de pågældende projektledere tildeler sager/projekter til de medarbejdere, som er tilknyttet deres projekter.
Arbejdsplanen kan dog kun klare at have et projekt pr dag for en medarbejder og skal justeres ofte. Desuden kan der opstå problemer, hvis projektlederne ikke snakker sammen om tildeling af ressourcer til deres projekter, eller en projektleder taster forkerte data ind i excel arket.



Figur 3 Excel ark for medarbejderbelastning

Det andet excel ark viser arbejdsbelastningen for hver enkel medarbejder. Det bruger data fra arbejdsplanen til at beregne projekttimerne for hvert projekt i de enkelte måneder, som medarbejderen er tildelt. På baggrund af disse timer beregnes en månedsbelastning for medarbejderen og 2 uddebiteringsgrader. Ulemperne ved dette excel ark er, at der stadigvæk skal laves nogle manuelle indtastninger i arket på nogle ting. Desuden skal arkets formler rettes af og til.

Dette er de 2 nuværende systemer, som bruges til at give et overblik over arbejdsplanlægningen i de enkelte afdelinger med hensyn til fordelingen af medarbejderressourcer på de enkelte projekter og deres arbejdsbelastninger.

1.2 Projekt formulering

På baggrund af de 2 eksisterende systemer ønskes et nyt webbaseret ressourceallokeringssystem til arbejdsplanlægning. Det nye system skal bruge elementer fra de 2 nuværende systemer, som findes nødvendige og bestå af 3 dele. For det første skal systemet bestå af en SQL server database, som indeholder de nødvendige oplysninger for arbejdsplanlægning herunder bl.a. data om medarbejdere, projekter/sager og relationer mellem disse..
Del 2 skal bestå af en web service, som skal tilgå databasen og indeholde de nødvendige rutiner/algoritmer for arbejdsplanlægning.
Den sidste del skal bestå af en webapplikation, som gør brug af de metoder, som web servicen tilbyder, og giver den grafiske brugerflade til at styre arbejdsplanlægningen i de repræsentative afdelinger. Systemet skal på sigt kunne bruges af alle afdelinger i Atkins Danmark.

1.3 Projekt afgrænsning

Det endelige resultat af dette 10 ugers eksamensprojekt vil være dækkende for den afdeling ”GIS & IT”, jeg arbejder i hos Atkins, da jeg gennem interviews med medarbejdere fra denne afdeling har fundet behovene til et nyt webbaseret ressourceallokeringssystem til arbejdsplanlægning i min indledende fase.
Prototypen på planlægningssystemet efter de 10 uger vil bestå af 3 dele.

- Del 1. En database på en SQL server, som indeholder muligheden for at opbevare alt den nødvendige data for arbejdsplanlægning. Herunder data for medarbejdere, sager/projekter og relationer mellem disse. Data vil bl.a. bestå af elementer fra de 2 nuværende systemer
- Del 2. En webservice, der har de nødvendige algoritmer/rutiner til brug for arbejdsplanlægning bl.a.
    1. Rutiner til oprettelse af projekter/sager og medarbejdere.
    2. Rutiner til Tildele/Slette/Opdatere projekter/sager på en medarbejders arbejdsplan.
    3. Rutine til visning af ressourceallokering(Arbejdsplan) for medarbejdere.
    4. Rutine til visning/beregning af arbejdsbelastning og uddebitering for den enkelte medarbejder
- Del 3. En meget simpel webapplikation laves, som gør brug af nogle af de implementerede web service metoder. Webapplikationen er i dette projekt kun beregnet til testvisning af brugen af den opbyggede web service og database.

1.4 Forkortelser

UML	-	Unified Modeling Language
UP	-	Unified procces
MVisio	-	Microsoft Visio 2003
DAAB	-	Data Access Application Block
WSDL	-	Web Service Description Language

1.5 Rapport skitse

Projekt rapporten er bygget op på følgende måde:

Resume

Giver en abstraktion over projektet

1. Introduktion

Her gives en introduktion til projektet med baggrund for projektet og hvad der vil blive fokuseret på i dette projekt.

2. Projekt planlægning

Her gives en introduktion til valg af udviklings proces i mit projekt og en plan for projektforløbet.

3. Brugerkrav specifikation

Her gives et overblik over funktionelle brugerkrav i form af use cases og ikke funktionelle brugerkrav for dette projekt og viser samtidig en risikoanalyse over use casene og en iterationsplan over de enkelte iterationer.

4. Analyse

Her gives en detaljeret beskrivelse af de enkelte use cases, samtidig vises en domain model over konceptet for systemet.

5. Design

Her vises sekvens diagrammer for de enkelte use cases og klasse diagram/web diagram for system, samtidig laves der et design for den mulige opbygning af databasen. Desuden diskuteres valg af design patterns.

6. Implementation

De implementerede software klasser beskrives, og de vigtigste metoder beskrives også, samtidig gives en beskrivelse af den implementerede database.

7. Test

Her gives en beskrivelse af de test, som er blevet brugt.

8. Konklusion

Her gives en konklusion på projekt diskussion med evaluering, perspektiv og fremtidige udvidelser.

9. Litteraturliste

Indeholder referencer på bøger og sider, som er brugt i dette projekt.

10. Bilag

Indeholder relevante diagrammer og dokumenter, som ikke er taget med i selve rapporten bl.a. Interviews, evaluerings dokument, og forskellige diagrammer.

Kapitel 2

Projekt planlægning

2. Projekt planlægning

2.1 Valg af udviklingsmetode

Til dette projekt har jeg valgt at benytte UP til udviklingen af mit projekt.
Jeg kunne selvfølgelig også have valgt andre udviklingsmetoder, men UP har en række fordele, som jeg vil kunne drage nytte af i mit projekt. Dette er bl.a.

Fordele:

- En iterativ og incremental udvikling
- Større fleksibilitet med hensyn til ændring af krav undervejs.
- Komplekse opgaver kan nemmere laves.
- Høj risiko emner kan identificeres tidligt og bearbejdes tidligt.
- Bedre erfaring undervejs efter hver iteration.

Da min projektperiode kun er på 10 uger, og alle kravene til systemet ikke er på plads fra starten af projektet, vil jeg med fordel kunne udnytte UP, da denne er mere fleksibel for ændringer af krav undervejs, samtidig vil man kunne se systemet vokse undervejs efter hver iteration.
Desuden vil jeg kunne identificere høj risiko emner fra starten af og bearbejde dem tidligt, såsom opbygningen af en database og de nødvendige cases, som skal laves.
Derudover giver den iterative udvikling mere erfaring for ens projekt undervejs, så man kan drage fordel af den i gennemførelsen af de næste iterationer.
Det skal dog siges, at UP også har sine ulemper, såsom

Ulemper:

- UML designet kan lave ting, som ikke altid kan laves praktisk i kode.
- Ændringer i de forskellige diagrammer kan tage tid.

Med hensyn til mit projekt er der dog flere fordele end ulemper ved at bruge UP til udvikling af projektet, så derfor har jeg valgt at benytte mig af denne.

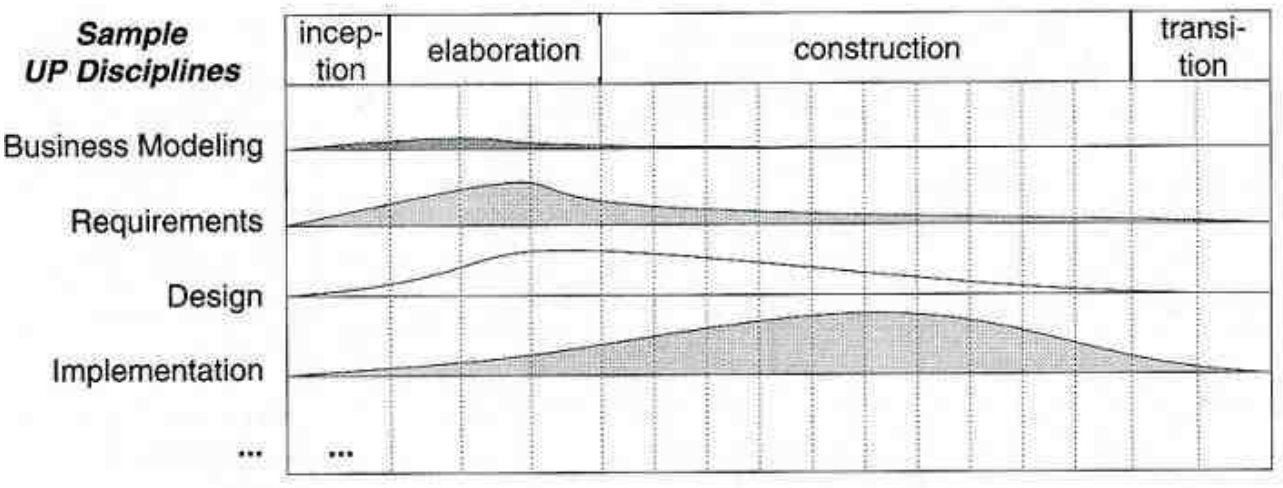
2.2 Unified Procces

UP (Unified Procces) er en velkendt software udviklingsproces, som er baseret på en iterativ udvikling og bruges meget bredt indenfor OOAD(objekt orienteret analyse og design). UP er ikke

en struktur, som skal følges slavisk, men en struktur brugerne kan tilpasse deres projekter. UP er en ”Best practices guide” man følger med en iterativ og risiko drevet udvikling.
UP gør brug af UML i de forskellige faser af udviklingsforløbet. Use casene bruges til at fange de funktionelle krav til systemet og definere indholdet i iterationerne.

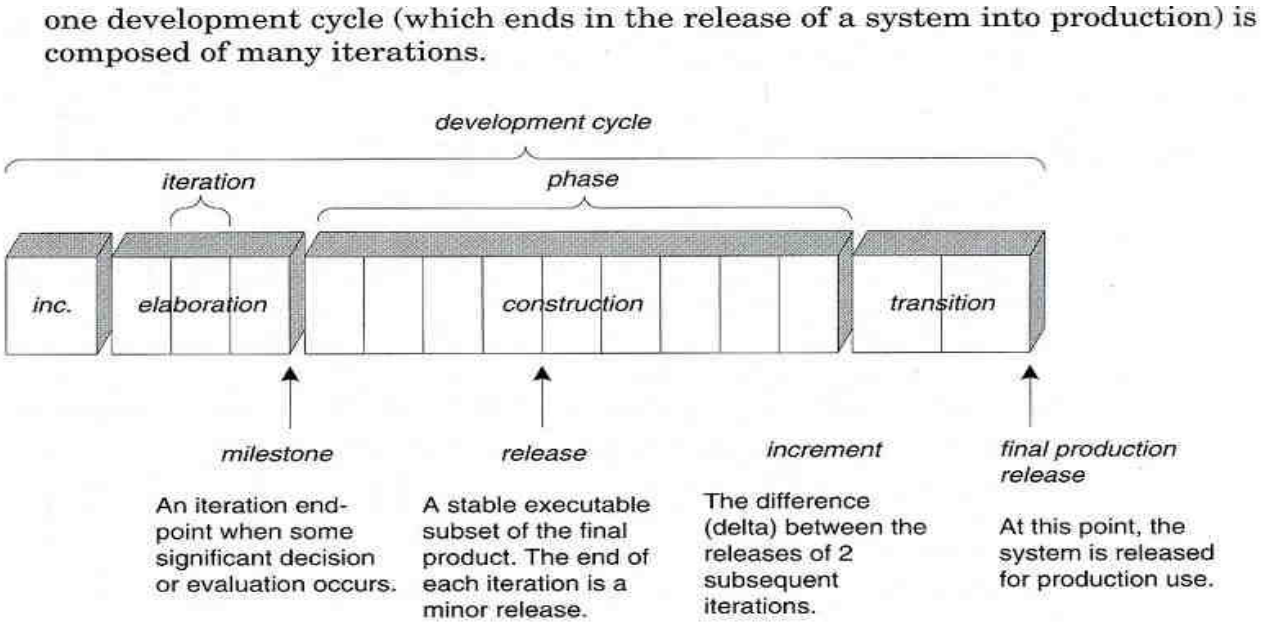
2.2.1 Unified Procces faser

En UP projekt organiserer arbejdet i iterationerne over 4 store faser[1, Side 33].



Figur 4 UP faser [1,36]

1. **En Inception fase** - Her laves en afgræsning for projektet, og use cases findes. Samtidig laves en estimation over risikoen for de enkelte use case. Der laves også en estimation over en plan for projektet.
2. **Elaboration –** Her redefineres afgræsningen af projektet. Der tages også fat på høj risiko emner, så man herved iterativt kan få implementeret kernearkitekturen for projektet.
   De fleste krav vil være fundet ved enden af denne fase og en mere nøjagtig estimation af projektplan kan laves.
3. **Construction -** Her tages fat i de sidste use cases med lavere risiko, og der forberedes på opsætning af systemet.
1. **Transition** – Denne fase bruges hovedsagligt til beta tests og opsætning af systemet. Det vil sige, her laves også bruger test af system, og små justeringer foretages. Efter denne fase er projektet klar til den sidste frigivelse.



Figur 5 UP udviklings cyklus [1,34]

Eftersom min projektperiode kun er på 10 uger, vil projektet ikke kunne nå gennem alle faser, men vil kun nå gennem Inception-, Elaboration- og Constructions fasen. På den anden side vil man dog stadigvæk have en prototype for systemet, eftersom systemets funktionalitet gradvis øges efter hver iteration. I mit projekt vil jeg ikke nå helt igennem constructions fasen, derfor vil nogle af kravene med lav risiko ikke blive lavet. Men de vil kunne gennemføres på et senere stadium, eftersom udviklingen fortsættes efter de 10 uger.

2.2.2 UML

Unified Modeling Language er et visuelt sprog til at modellere strukturer og forløb i primært objekt orienteret softwaresystemet. Jeg vil igennem mit projektforløb gøre brug af følgende UML diagrammer.

- **Use case** - Bruges til at beskrive, hvordan en aktør bruger systemet.
- **System sekvens** - Bruges til at vise, hvordan en aktør kommunikere med systemet.
- **Domain model** – En domain model er en visualisering af virkelighedens begreber, som man prøver at lave et system for.
- **Klasse** – Bruges til at beskrive, hvilke egenskaber og metoder de enkelte klasser indeholder, samtidig med at det beskriver hvilken associationer, der er imellem de enkelte klasser.

- **Sekvens** – Viser interaktion mellem klasserne i systemet med hensyn til metodekald.

Desuden vil jeg til opbygning af en database benytte følgende diagram.

- **ER diagram** - Et ER diagram er en datamodel af en database.

For ikke at gå for meget i detaljer med diagrammerne under udviklingsforløbet, vil ikke alle UML notationer på de enkelte diagrammer blive brugt, men kun de notationer, som findes nødvendige på dette projekt. Desuden er der i det værktøj MVisio, som jeg bruger, ikke altid mulighed for eller praktisk at lave visse notationer, så der vil jeg lave notationer i en simplere udgave.

2.3 Projektplan

Projektplan blev lavet i starten af mit projekt for at have noget at arbejde efter og for at danne mig et overblik over, hvordan projektet kunne forløbe.
Det er ikke meningen, at projektplanen skal følges 100 %, som den er beskrevet, da det på så tidligt et stadium i projektet ikke er muligt at vide præcist, hvordan forløbet bliver. I stedet for er der i brugerkrav specifikationen lavet en iterationsplan[se afsnit 3.4], som blev lavet løbende, efterhånden som iterationer blev færdige, og en ny blev planlagt.

Faser i OOAD	Iteration	Handling	Dato	Uge
User requirements	Inception	Identifikation af Use Cases og aktører. Der laves en række interviews med valgte aktører for at finde de relevante use case, som der skal arbejdes med. Risiko analyse laves over use casene. Iterations plan laves.	6 -10. August	1
Krav Analyse	1 iteration / Elaboration	Beskrivelse af use case laves. Beskrivelse af dem med størst risiko laves udførligt. Samtidig laves en domain model over konceptet for systemet. ER diagram laves over konceptet for database.	13 -17. August	2
Krav Analyse / Design / Implementation / Test	1 iteration / Elaboration	Relationsdiagram laves og databasen opbygges. Samtidig laves klassediagram og sekvens diagram for valgte use cases. Prototype implementeres herefter over valgte use cases. I slutningen laves en test over færdig prototype.	20 – 24. August	2-3
Krav Analyse / Design / Implementation / Test	2 iteration	Erfaring for 1 iteration bruges til nyt design i 2 iteration og til at opdatere design for 1 iteration. Herefter fortsættes ligesom i 1 iteration med at lave use case beskrivelse / sekvens diagrammer / klassediagram / implementation for de næste use cases.	27-31. August	4
Krav Analyse / Design / Implementation / Test	2 iteration	Der forsættes med at lave Implementation og test for valgte use cases.	3-7. September	5
Krav Analyse / Design / Implementation / Test	3 iteration	Erfaring for 2 iteration bruges til nyt design i 3 iteration og til at opdatere design for 2 iteration. Herefter startes på de sidste vigtigste use cases og prototypen udvides.	10-14. September	6
Design/ Implementation /Test	4 iteration	Erfaring for 3 iteration bruges til nyt design i 4 iteration og til at opdatere design for 3 iteration. Herefter laves de lette use cases ligesom i de andre iterationer.	17-21. September	7
Afslutning	5 iteration	De sidste små justeringer laves / Rapport	24-28. September	8
Afslutning	6 iteration	Rapport	1 – 5. Oktober	9
Afslutning	6 iteration	Rapport	8 – 12. Oktober	10
Aflevering	6 iteration	Rapport afleveres	15. Oktober	11

Møder	Iteration	Mål med møde	Dato	Uge
Første møde	1 iteration	Snak om projekt med nye ændringer i forhold til beskrivelsen bilag 8 og snak om rapportens indhold.	15 / 8-2007	2
Kollokvium	4 iteration	Snak med DTU Vejleder/Atkins vejleder om projekt status og plan for sidste del.	20 / 9-2007	7

2.4 Resume

I planlægningen fik jeg begrundet mit valg af udviklingsmetoden UP. Jeg fik samtidig forklaret lidt om, hvordan den kører, og hvordan jeg vil bruge den. Til sidst fik jeg lavet en estimation over projektplanen for de 10 uger, projektet vil vare.

Kapitel 3

Brugerkrav specifikation

3 Brugerkrav specifikation

3.1 System krav

For at finde kravene til systemet, deles de op i 2 dele, de funktionelle og de ikke funktionelle.
De funktionelle krav beskriver brugerens krav til systemets funktionalitet, og hvordan systemet skal bruges, mens de ikke funktionelle krav indeholder alle andre krav til systemet, såsom understøttelse af andre systemer. For at finde de funktionelle krav til systemet har jeg lavet en række små interviews[se bilag A] med 3 personer, som hver især passer på en af de 3 aktører, jeg har fundet til systemet, i afdelingen som jeg arbejder i. De funktionelle krav til systemet beskrives vha. Use Case modellen[se afsnit 3.3]. De ikke funktionelle krav beskrives under punkt ”3.2 Supplerende system krav”.

3.2 Supplerende system krav

3.2.1 Brugbarhed

Systemet skal gøre det lettere at få et overblik over arbejdsplanlægning med hensyn til, hvad medarbejdere arbejder med..
Når den fulde version af webapplikationen på et tidspunkt bliver lavet, skal visse funktioner ikke være tilgængelige for alle aktører, bl.a. skal mulighederne for at tildele/opdatere/slette sager fra en medarbejders arbejdsplan kun kunne gøres af en projektleder.

3.2.2 Driftsikkerhed

Systemet skal levere korrekte data og være nogenlunde stabilt med hensyn til visning af en medarbejders arbejdsbelastning og arbejdsplan for en afdeling, mens driftsikkerheden især for tildeling/opdatering/sletning af sager på en medarbejders arbejdsplan skal foregå korrekt. F.eks. skal tildelingen af en sag på en dato ikke konflikte med en allerede tildelt sag.
Hvis en ting ikke kan lade sig gøre, skal der gives en sigende fejlmeddelelse.

3.2.3 Præstation

Systemets svartider skal være indenfor en grænse på max 1 min med hensyn til en given forespørgsel.

3.2.4 Understøttelse af andre systemer

Systemet skal som udgangspunkt bruge en SQL server database, men skal laves så det let kan ændres til en Oracle database uden at ændre for meget i koden. Derfor skal DAAB, under Microsoft

Enterprise library, anvendes til databasen, hvilket vil gøre det let at skifte til database type uden at ændre for meget i koden. For at understøtte muligheden for opbygning af flere forskellige webapplikationer ønskes en web service lavet, som laver alt arbejde med databasen og leverer de nødvendige metoder til brug for de forskellige webapplikationer.

3.2.5 Implementation

Til implementation af denne eksamensopgave benyttes værktøjerne Microsoft Visual Studio 2005 professionel udgaven med .net og c# til at lave web servicen og testwebapplikationen.
Til databasedelen vil Microsoft SQL server 2005 blive brugt til databasen, og Microsoft Enterprise library januar 2006's DAAB vil blive brugt for tilgang til databasen.
Databasen skal indeholde de elementer, som findes nødvendige fra de 2 nuværende systemer.

3.2.6 Test

Til testning af dette projekt skal Nunit test bruges på nogle af de vigtigste metoder i web service del. Desuden skal der efter hver iteration laves en manuel test af systemet.

3.3 Use case model

3.3.1 Identifikation af aktører

Til brug af arbejdsplanlægningssystemet har jeg fundet 3 aktører, som har interesse i sådan et system. Den første aktør er en almindelig arbejder, som gerne vil have styr over, hvad han/hun skal arbejde med frem i tiden.
Den anden aktør består af en projektleder, som gerne vil have overblik over, hvad medarbejderne i afdelingerne laver, og hvad de enkelte medarbejders arbejdsbelastninger er, så han/hun kan finde ressourcer til sine projekter, da han/hun gerne vil have ressourcer nok til at få sine projekter gennemført. Overblikket kan give projektlederen mulighed for at tildele en medarbejder til et af sine projekter, hvis de er til rådighed.
Den sidste aktør består af en afdelingsleder/chef, som gerne vil have et overblik over timerne på projekterne og benyttede medarbejderes kapaciteter med hensyn til deres arbejdsbelastninger og uddebiteringsgrad.

3.3.2 Identifikation af use cases

For at finde ud af, hvad de 3 fundne aktører har af brugerbehov til sådan et system, har jeg valgt at lave et interview med en fra hver af de 3 grupper, en afdelingsleder, en projektleder og en medarbejder i min afdeling. Hvis det havde været et længere projekt vil det selvfølgelig have

involveret flere personer. [Se bilag A for interviews]. Ud fra de 3 interviews og generelle behov har jeg fundet frem til følgende use cases, som der skal arbejdes ud fra.

Generelle for alle

- Find medarbejder/medarbejdere.
  Giver muligheden for at finde en medarbejder eller alle medarbejdere for en afdeling.
- Find afdelinger.
  Bruges til at finde alle afdelinger.

Medarbejder:

- Vis min arbejdsplan.
  Bruges til at vise arbejdsplan for en medarbejder.
- Vis arbejderbelastning for medarbejder.
  Skal give mulighed for at vise arbejdsbelastningen for en medarbejder.
- Vis arbejdsplan for medarbejdere.
  Bruges til at vise arbejdsplanen for alle medarbejdere i en afdeling.

Afdelingsleder/Chefer:

- Find projekter/sager
  Giver mulighed for at finde alle projekter eller sager.
- Opret organisation
  Giver mulighed for at oprette en ny sektor, afdeling eller underafdeling for en organisation.
- Vis benyttede medarbejder kapaciteter.
- Vis timeregnskab for projekter.
  Bruges til at give et overblik over benyttede timer for projekter med hensyn til medarbejdere, som er tildelt projekterne.
- Vis frie ressourcer.
  Skal kunne vise medarbejdere, som har timer til rådighed i perioder.

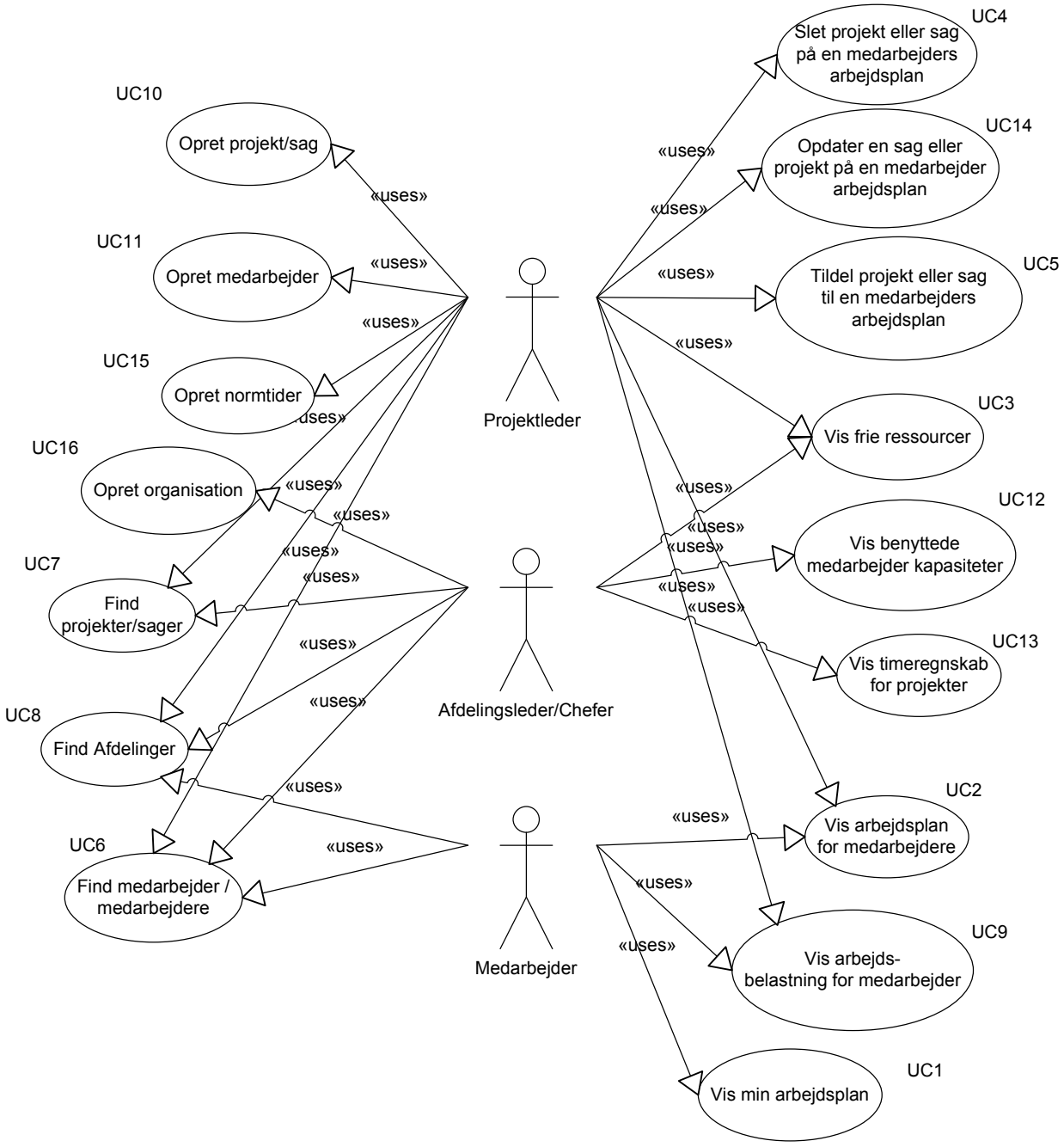
Projektleder:

- Find projekter/sager
  Giver mulighed for at finde alle projekter eller sager.
- Vis frie ressourcer
  Skal kunne vise medarbejdere, som har timer til rådighed i perioder.
- Vis arbejdsbelastning for medarbejder
  Skal give mulighed for at vise arbejdsbelastningen for en medarbejder.

- Vis arbejdsplan for medarbejdere.
  Bruges til at vise arbejdsplanen for alle medarbejdere i en afdeling.
- Tildel projekt og sag til en medarbejders arbejdsplan.
  Giver muligheden for at tildele et projekt eller en sag for en periode til en medarbejder.
- Slet projekt eller sag på en medarbejders arbejdsplan.
  Giver muligheden for at slette et projekt eller sag, som er tildelt en medarbejder.
- Opdater en sag eller projekt på en medarbejders arbejdsplan.
  Giver muligheden for at ændre en allerede tildelt sag til en anden.
- Opret normtider
  Giver muligheden for at indtaste antal arbejdsdage for en måned og år.
- Opret medarbejder
  Giver mulighed for at oprette nye medarbejdere i systemet med den nødvendige information
- Opret projekt/sag
  Bruges til at oprette nye projekter og sager i systemet

De fundne use case fra interviews og generelle ønsker/behov vil blive bearbejdet efter en risiko analyse af de enkelte use cases[se afsnit 3.3.4].
Herved kan de vigtigste use cases for systemet blive gennemarbejdet først, da jeg sandsynligvis ikke når at lave alle use case.
Formålet med opgaven var også at finde generelle behov til sådan et planlægningssystem, ikke at implementere alt i projektperioden.
De manglende behov vil der senere kunne arbejdes videre med efter projektperioden, og nye vil stadigvæk kunne nås at blive tilføjet, da UP metoden netop bygger på, at det skal kunne lade sig gøre at tilføje nye krav undervejs. Det er netop ikke meningen med UP, at man skal finde alle behovene fra starten.

3.3.3 Use case diagram



Figur 6 Use case diagram.

3.3.4 Use case risiko analyse

Use casene skal bearbejdes, således at dem med højest prioritet laves først i elaborations fasen, derfor laver jeg en risiko vurdering af de enkelte use cases.
Prioriteten beregnes vha. følgende formel.

*Prioritet(0-100) = Konsekvens grad * Kompleksitets grad * Risiko grad*

Konsekvens grad (0-1) siger noget om den konsekvens, den enkelte use case har for systemets virkning. Dvs. hvis det f.eks. er en fed feature, som man gerne vil have til system, men som ikke har nogen betydning for systemet, fås en konsekvens grad på 0.
Kompleksitets grad (0-10) siger noget om sværhedsgrad af en implementering og egen erfaring med opgaven.
Risiko grad (0-10) siger noget om risiko for implementering af den enkelte use case, i forhold til at systemet er meget afhængig af, at denne use case virker.

Use case Id	**Navn**	**Konsekvens-grad (0-1)**	**Kompleksitets-grad (0.-10)**	**Risiko-grad (0-10)**	**Prioritet (0 -100)**	**Kommentar**
6	Find medarbejder / medarbejdere.	1	1	5	5	Denne use case er relativ simpel at lave. Den er i første omgang beregnet til at blive brugt sammen med UC 4,UC 5 UC 14 og UC 9 i webapplikationen..
##	Databasen	1	8	10	80	Er den vigtigste for systemet. Skal derfor laves som det første.
1	Vis min arbejdsplan.	1	4	2	8	Er ikke så vigtig i første omgang, da UC2 kan blive brugt i forløbet..
8	Find Afdelinger	1	1	5	5	Denne use case er relativ simpel at lave. Den er i første omgang beregnet til at blive brugt sammen med UC 2 i webapplikation..
7	Find projekter / sager	1	1	5	5	Denne use case er relativ simpel at lave. Den er i først omgang beregnet til at blive brugt sammen

						med UC 4, UC 5, UC 14.
2	Vis arbejdsplan for medarbejdere.	1	7	10	70	Denne use case er meget vigtig for systemet, da den giver et overblik over arbejdsplanen for en given afdelings medarbejdere.
3	Vis frie ressourcer	0	5	1	0	Use case er udgået fra opgaven, i og med at man kan se frie medarbejdere ressourcer i UC 2 og UC 9.
5	Tildel projekt eller sag til en medarbejders arbejdsplan.	1	8	8	64	Denne use case er vigtig for systemet, eftersom det ellers ikke vil være noget data på arbejdsplan for medarbejderne.
4	Slet projekt eller sag på en medarbejders arbejdsplan	1	8	8	64	Denne use case er vigtig for systemet eftersom det ellers ikke vil være muligt at slette tildelte sager på en medarbejderes arbejdsplan.
11	Opret medarbejder	1	1	10	10	Er forholdsvis let at lave, men vil være vigtig for systemets virkning.
10	Opret projekt/sag	1	1	10	10	Er forholdsvis let at lave, men vil være vigtig for systemets virkning.
12	Vis benyttede medarbejder kapaciteter	0	8	1	0	Laves senere da dette er en feature, som allerede er i systemet blast 32, men vil være god at få med senere.
13	Vis timeregnskab for projekt	0	7	1	0	Dette beregnes på nuværende tidspunkt i økonomisystemet SAP. Men da økonomisystemet ikke beregner frem i tiden, kunne det være rart, hvis planlægningssystemet

						kunne lave timeberegninger på et projekt frem i tiden, eftersom medarbejdere er tildelt projekter perioder frem i tiden.
9	Vis arbejds-belastning for medarbejder	1	8	6	48	Denne use case er vigtig, da den kan laver en prognose 1 år frem i tiden med arbejdsbelastninger og uddebiteringsgrader for hver måned for en medarbejder.
14	Opdater en sag eller projekt på en medarbejders arbejdsplan.	1	8	6	48	Denne use case er vigtig for systemet, eftersom det ellers ikke vil være muligt at ændre på en medarbejders arbejdsplan. Det skal dog siges, at virkningen er mindre, eftersom en ændring kan klares med en sletning og ny tildeling.
15	Opret normtider	1	1	10	10	Selvom prioritet af denne ikke er så høj, vil denne use case være nyttig at lave samtidig med UC 9, da den er forholdsvis nem at lave og vigtig for systemets virkning..
16	Opret organisation	1	1	5	5	Denne use case er forholdsvis simpel at lave, og vil først senere være nødvendig i og med, at organisation i databasen kan oprettes manuelt i starten.

3.4 Iterations plan

Iterations planen bruges til at beskrive, hvad der skal arbejdes med i de enkelte iterationer. Planen for de enkelte iterationer laves løbende, da man efter hver iteration planlægger den næste med erfaringer fra den sidste. Herved opbygger man større erfaring undervejs i processen.
I starten af hver iteration laves de ændringer, som er fundet nødvendige på baggrund af erfaring fra den forrige iteration.

Iteration	**Mål**	**Use cases der arbejdes med**	**Periode**
Inception fase	Få identificeret use cases via interviews(med aktører) og generelle behov, samtidig findes andre funktionaliteter. Risiko analyse laves over use casene og en projektplan formuleres.		6 – 10. August
1. iteration / Elaboration	Få lavet use case beskrivelserne. UC2 er valgt til denne iteration, eftersom den er vigtigst for systemet, derfor laves der en udførlig beskrivelse af den. Bagefter laves design og implementation af den. Samtidig med at der for den påtænkte database laves et ER diagram, som databasen bagefter oprettes ud fra. Ved slutning af denne iteration laves en test af prototypen. Erfaringer fra denne iteration bruges til planlægningen af iteration 2.	UC2 og database	13 – 24. August
Evaluering over 1. iteration	Se bilag B for beskrivelsen af evalueringen af 1. iteration.		
2. iteration / Elaboration	I iteration 2 vælges de næste use cases efter prioriteten, som der skal arbejdes med. Der bliver for de valgte use cases lavet en udførlig use case beskrivelse og design/implementation af dem ligesom i	UC4, UC5 og UC14	27 – 31. August

	iteration 1. Til sidst i iteration testes de nye funktioner. Erfaringer fra denne iteration bruges til planlægning af iteration 3.		
Evaluering over 2. iteration	Se bilag B for beskrivelsen af evalueringen af 2. iteration.		
3. iteration / Elaboration	I iteration 3 laves de sidste use cases med høj risiko ligesom i de forrige iterationer. Til sidst i iterationen testes de nye funktioner. Erfaringer fra denne iteration bruges til planlægningen af iteration 4.	UC 15, UC9	3 – 7. September
Evaluering over 3. iteration	Se bilag B for beskrivelsen af evalueringen af 3. iteration.		
4. iteration / Construction	I iteration 4 arbejdes der med de lette og lav risiko use cases, da de vigtigste er bearbejdet i de forrige. Til sidst i denne iterationen testes de nye funktioner. Erfaringer fra denne iteration bruges til planlægningen af iteration 5.	UC 10, UC 11, UC 1, UC 8	10-14. September
Evaluering over 4. iteration	Se bilag B for beskrivelsen af evalueringen af 4. iteration.		
5. iteration / Construction	I iteration 5 arbejdes der med de sidste lavrisiko use cases, som kan nås i dette projekt, samtidig laves der nogle mere generelle test. Til sidst i denne iterationen testes de nye funktioner. Erfaringer fra denne iteration bruges til planlægningen af iteration 6.	UC 6, UC 7, UC 16	17-21. September
Evaluering over 5. iteration	Se bilag B for beskrivelsen af evalueringen af 5. iteration.		

6. iteration	Denne iteration vil udelukkede blive brugt til rapport, da det er det vigtigste på dette tidspunkt.	Rapport	24-10. September – Oktober

3.5 Resume

I brugerkrav specifikationen fandt jeg de 3 aktører, som kunne have gavn af at benytte planlægningssystemet, og de tilhørende use cases. Samtidig fandt jeg andre krav til systemet, som ikke kunne beskrives vha. use cases. Derefter blev der lavet en risikoplan over de fundne use cases, hvor det blev fastlagt, i hvilken rækkefølge de skulle bearbejdes. Herefter blev en begyndende iterations plan lavet. Denne blev opdateret løbende, eftersom iterationerne blev færdige og nye planlagt.

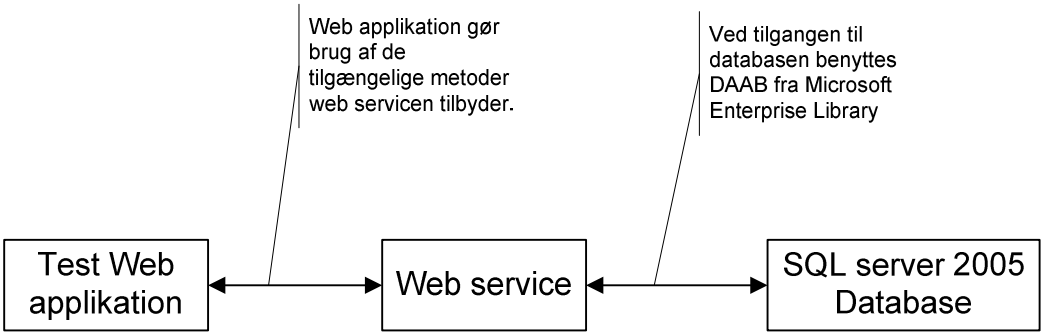
Kapitel 4

Analyse

4 Analyse

4.1 Opbygning af system

Ud fra de stillede supplerende krav til systemet [se afsnit 3.2] og opgaveafgrænsningen har jeg forestillet mig systemet bygget op på følgende måde.



Figur 7 Opbygning af planlægningssystem.

Herved understøttes kravene til, at databasen nemt skal kunne udskiftes, og at de tilgængelige metoder web servicen tilbyder, kan bruges af flere webapplikationer. Desuden vil tilgangen til databasen blive nemmere, da meget af den redundante kodning bliver reduceret ved inkludering af DAAB, som indeholder klasser, der bl.a. sørger for at lave connection objekter og åbner forbindelser.

4.2 Use cases og system sekvens diagrammer

Use case beskrivelsen giver en beskrivelse af, hvad formålet er med enkelte use case, og hvordan en aktør bruger den pågældende use case, samt hvilke kritikere den har.
System sekvens diagram bruges til at vise brugerens interaktioner med systemet.
De use cases jeg fandt under afsnit 3.3.2 vil under dette punkt blive beskrevet mere specifikt med en use case beskrivelse og et system sekvens diagram. Dog vil kun de vigtigste use cases og diagrammer fremgå under dette punkt. De mindre vigtige vil kunne ses i bilag C.

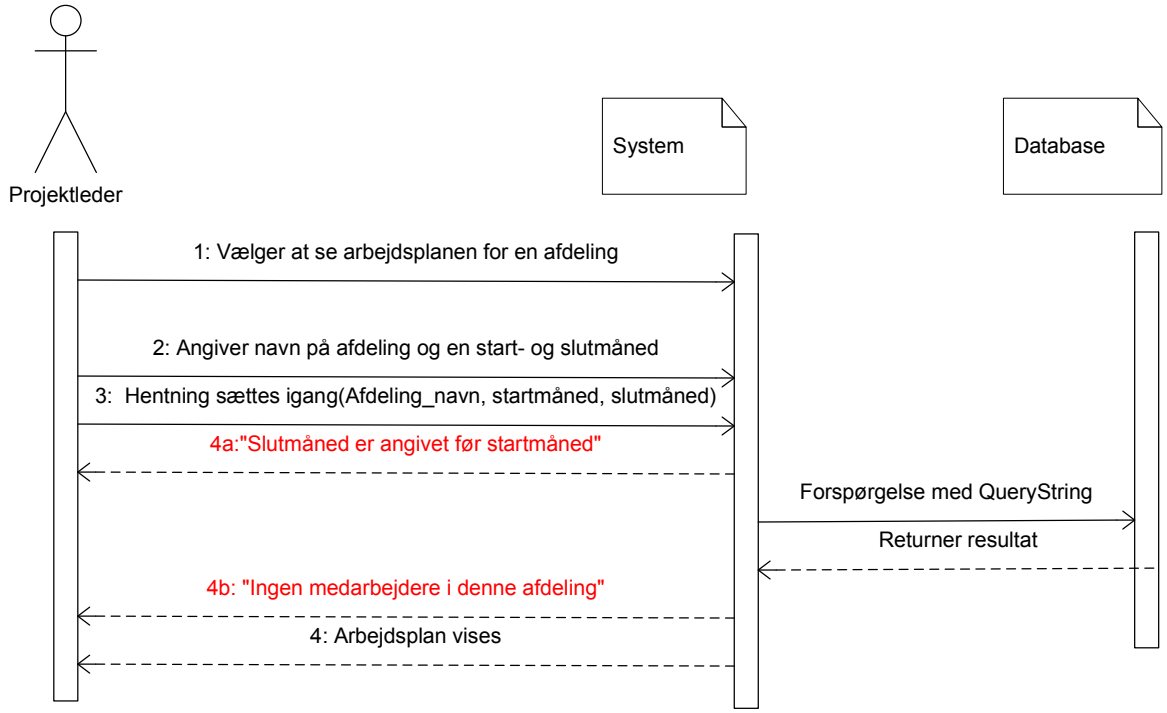
4.2.1 UC 2 Vis arbejdsplan for medarbejdere

Use case ID	UC 2	**Use case navn**	Vis arbejdsplan for medarbejdere.
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	**09. August 2007**	**Sidste opdaterings dato**	**20. August 2007**
Beskrivelse af mål med use case	Målet er at få et overblik over, hvad medarbejdere i en given afdeling er allokeret på projekter/sager for en måned eller flere måneder frem i tiden. Dog max 1 år.		
Forudsætninger	UC 8 er blevet kørt, hvor ved alle afdelinger findes.		
Succes kondition	Projektlederen får et overblik over, hvem der arbejder med hvad i en måned eller for flere måneder frem i tiden. Dog max 1 år.		
Fejl kondition	Projektlederen får ikke et overblik over medarbejdernes arbejdsplaner.		
Primær Aktør	Projektleder		
Sekundær Aktører	Afdelingsleder/Chefer og Medarbejder.		
Trigger	Projektleder ønsker at se arbejdsplanen for medarbejderne i en afdeling.		
Normalt forløb	1. Brugeren vælger at se arbejdsplanen for en afdeling. 2. Brugeren angiver en afdeling, en periode med start- og slutmåned, denne ønsker at se en arbejdsplan for. 3. Brugeren vælger at hente arbejdsplanen for den valgte periode og afdeling. 4. Systemet returnerer en arbejdsplan og viser den med information om tildelte projekter/sager for alle medarbejdere i den valgte afdeling. Arbejdsplanen viser en måned af gangen og giver brugeren mulighed for at gå måneder frem og tilbage. Den enkelte måned vises med hver arbejdsdag for en medarbejder delt op i formiddag og eftermiddag.		
Alternativt forløb	**Step**	**Branching Action**	
	4a	Hvis slut måned er angivet før startmåned gives en fejlmeddelelse ”Slutmåned er angivet før startmåned”	

	4b	Hvis der ikke kunne findes nogen medarbejder i den valgte afdeling returneres en fejlmeddelelse ”Ingen medarbejder i denne afdeling”

Beslægtet information	Denne use case vil kunne bruges på sammen måde af afdelingsleder og medarbejder.
Prioritet:	70 ud af 100 meget vigtig for projektet
Udførelse	Mindre end 1 minut.
Hyppighed	Bruges regelmæssigt.
Åbne spørgsmål	Webapplikationen vil i dette projekt kun være en simpel test version, så selve måden hvorpå man vælger flere parametere skal måske specificeres senere hen.
Afslutnings dato	24. August 2007

System sekvens diagram



Figur 8 - System sekvens diagram UC 2

4.2.2 UC 5 Tildel projekt eller sag til en medarbejders arbejdsplan

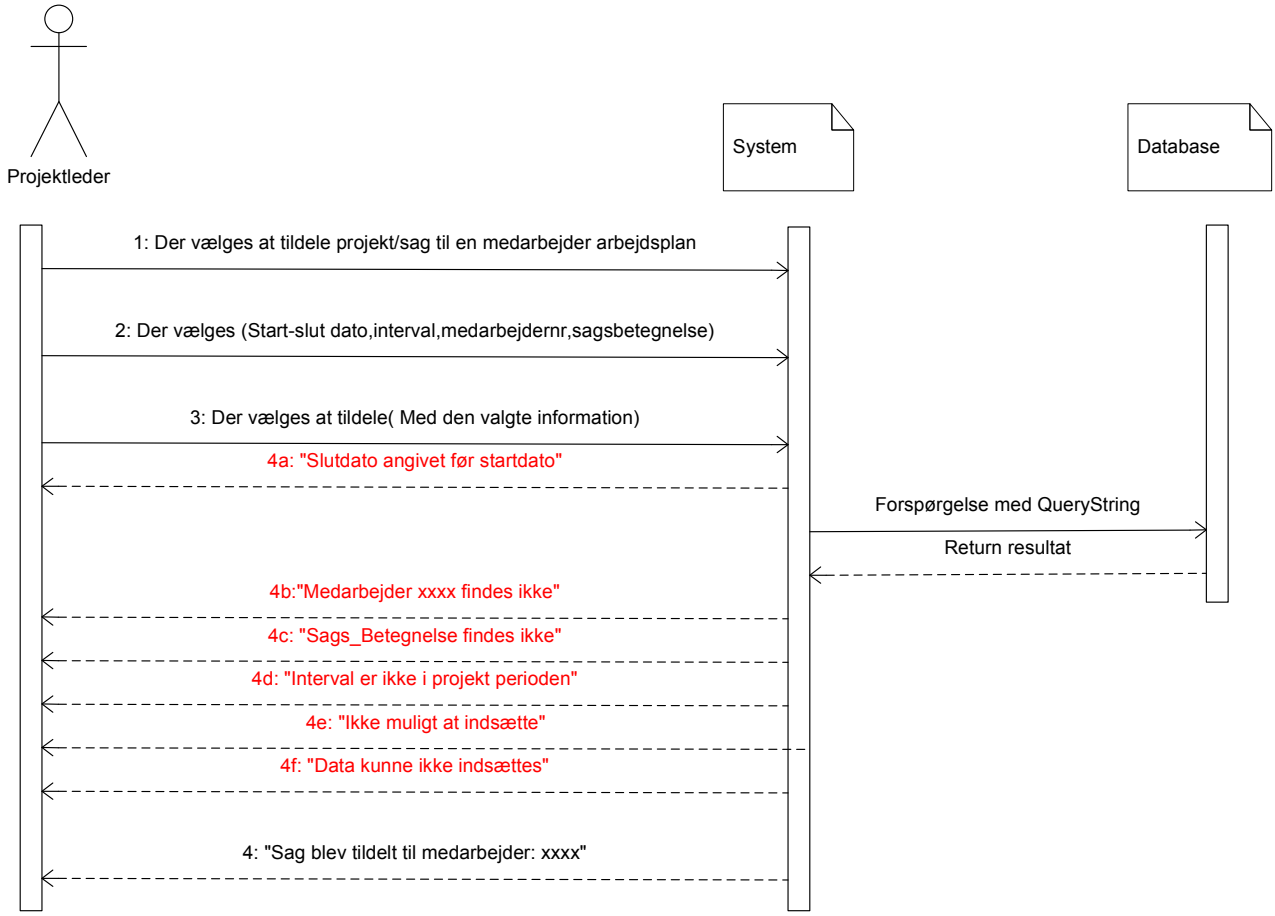
Use case ID	UC 5	**Use case navn**	Tildel projekt eller sag til en medarbejders arbejdsplan.
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	**09. August 2007**	**Sidste opdaterings dato**	**27. August 2007**
Beskrivelse af mål med use case	Der ønskes at tildele en medarbejder et projekt eller sag(Ferie, Barsel) for en given periode på medarbejderens arbejdsplan med en opdeling på, om der arbejdes formiddag, eftermiddag eller hele dagen i perioden. Man skal ikke kunne tildeles noget på hellig dage eller i weekender, samtidig skal der tages hensyn til skudår.		
Forudsætninger	UC 6 og UC 7 er blevet kørt først. Hvor ved alle medarbejders initialer findes for en specifik afdeling(På dette tidspunkt GIS & IT).), og alle sagsbetegnelser for projekter og sager findes.		
Succes kondition	En medarbejder får tildelt et projekt eller en sag(Ferie, Barsel) i den givne periode med en vægt i dennes arbejdsplan.		
Fejl kondition	Medarbejder bliver ikke tildelt noget i sin arbejdsplan.		
Primær Aktør	Projektleder		
Trigger	En medarbejder skal tildeles et projekt eller sag(Barsel, ferie, Etc) i dennes arbejdsplan..		

Normalt forløb	1. Brugeren vælger at tildele en medarbejder et projekt/sag på dennes arbejdsplan. 2. Projektlederen vælger følgende for at tildele en sag/ projekt til medarbejderens arbejdsplan: • En periode med start- og slutdato. • Et interval(Formiddag, Eftermiddag eller hele dagen). • Medarbejder initialer. • En sags betegnelse for et projekt eller sag. 3. Projektlederen vælger at tildele ud fra de valgte oplysninger. 4. System returnerer meddelelsen ”Sag blev tildelt til medarbejder: xxxx” nå tildelingen lykkes.	
Alternativt forløb	**Step**	**Branching Action**
	4a	System returnerer fejlmeddelelsen ”Slutdato angivet før startdato”, hvis slutdato er angivet før startdato.
	4b	System returnerer fejlmeddelelsen ”Medarbejder xxxx findes ikke”, hvis der ikke findes nogen medarbejder med de initialer.
	4c	System returnerer fejlmeddelelsen ”Sags_betegnelse findes ikke”, hvis sagsbetegnelsen ikke findes.
	4d	I det tilfælde man tildeler et projekt, returnerer systemet en fejlmeddelelse ”Interval er ikke i projekt perioden”, hvis den angivne periode ligger udover et projekts start- og slutperiode.
	4e	System returnerer meddelelsen ”Ikke muligt at indsætte” hvis det ikke kunne lade sig gøre at indsætte, pga. at medarbejderen allerede er tildelt noget i den angivne periode og interval.
	4f	System returnerer meddelelsen ”Data kunne ikke indsættes”, hvis der sker en fejl ved indsættelsen.

Beslægtet information	For at se om det er muligt at tildele noget til medarbejders arbejdsplan i perioden henføres til use case UC2.
Prioritet	64 ud af 100 meget vigtig for projektet
Udførelse	Max 1 minut.
Hyppighed	Benyttes dagligt

Åbne spørgsmål	Web applikationen vil i dette projekt kun være en simpel test version, så selve måden hvorpå man vælger parametere skal måske udbygges senere hen.
Afslutnings dato	31. August

System sekvens diagram



Figur 9 – System sekvens diagram UC 5

4.2.3 UC 4 Slet projekt eller sag på en medarbejders arbejdsplan

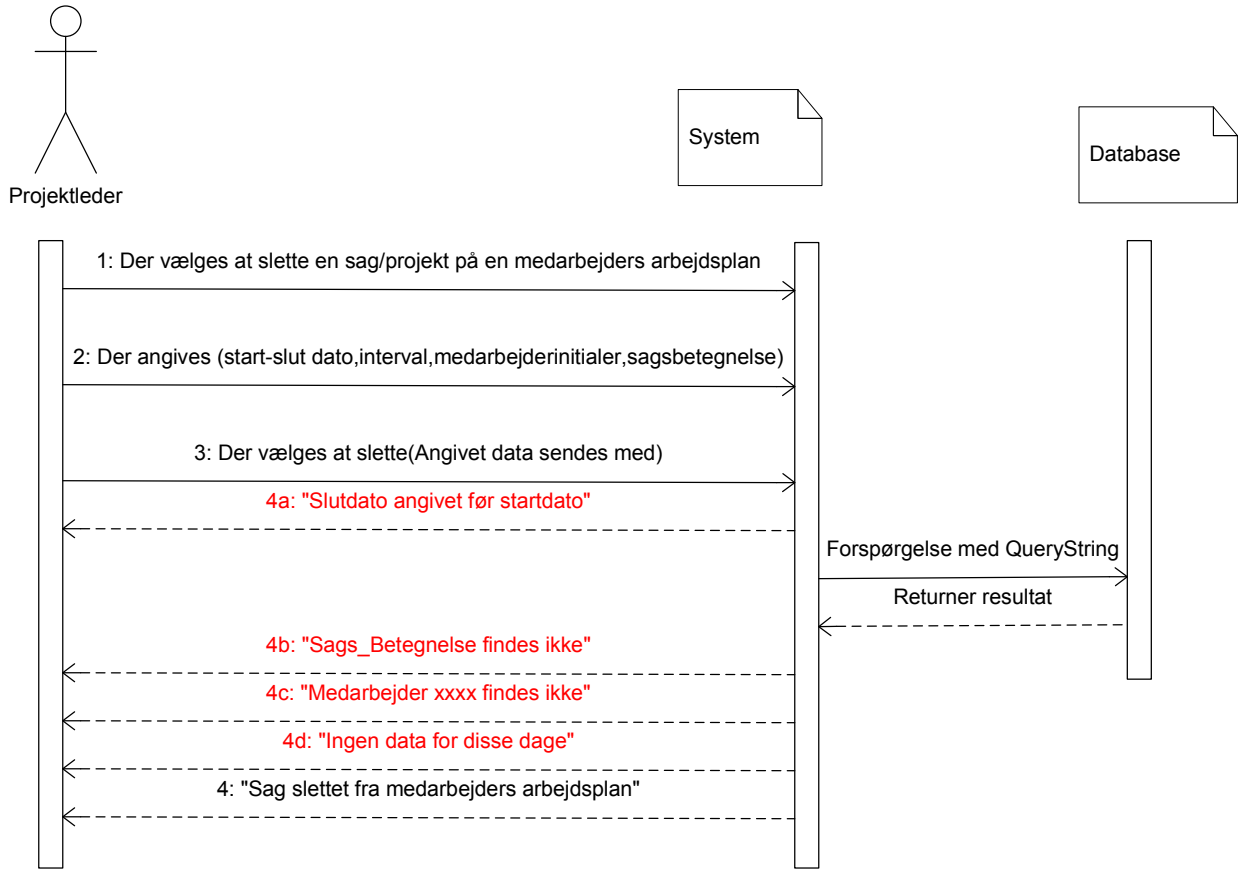
Use case ID	UC 4	**Use case navn**	Slet projekt eller sag på en medarbejders arbejdsplan.
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	**24. August 2007**	**Sidste opdaterings**	**28. August 2007**

		dato	
Beskrivelse af mål med use case	Der ønskes at slette en sag eller et projekt på en medarbejders arbejdsplan for en periode med interval på, om det skal være om formiddagen, eftermiddagen eller hele dagen i perioden.		
Forudsætninger	UC 6 og UC 7 er blevet kørt først. Hvorved alle medarbejders initialer findes for en specifik afdeling (På dette tidspunkt GIS & IT)., og alle sagsbetegnelser for projekter og sager findes.		
Succes kondition	Projekt eller sag bliver slettet for en medarbejders arbejdsplan i den angivne periode og interval.		
Fejl kondition	Projekt eller sag kunne ikke slettes på medarbejderens arbejdsplan.		
Primær Aktør	Projektleder		
Trigger	En projektleder ønsker at slette et projekt/sag på en medarbejders arbejdsplan.		
Beskrivelse	**Step**	**Aktion**	
Normalt forløb	1. Brugeren vælger at slette en sag/projekt på en medarbejders arbejdsplan. 2. Brugeren vælger følgende for at slette en sag/ projekt på medarbejderens arbejdsplan: • En periode med start- og slutdato. • Et interval(Formiddag, Eftermiddag eller hele dagen). • Medarbejder initialer.. • En sags betegnelse for projektet eller sagen, som skal slettes. 3. Brugeren vælger at slette ud fra de valgte oplysninger. 4. Systemet returnerer meddelelsen ”Sag slettet fra medarbejders arbejdsplan” når sletningen kunne gennemføres.		
Alternativt forløb	**Step**	**Branching Aktion**	
	4a	System returnerer fejlmeddelelsen ”Slutdato angivet før startdato”, hvis slutdato er angivet før startdato.	
	4b	System returnerer fejlmeddelelsen ”Sags_Betegnelse findes ikke” hvis sagsbetegnelsen ikke findes.	
	4c	System returnerer fejlmeddelelsen ”Medarbejder xxxx findes ikke” hvis medarbejderen med de initialer ikke findes.	
	4d	System returnerer fejlmeddelelsen ”Ingen data for disse dage”, hvis	

		der ingen data er på i den valgte periode og interval.

Beslægtet information	For at se om det er muligt at slette noget fra en medarbejders arbejdsplan i perioden henføres til use case UC2.
Prioritet	64 ud af 100 meget vigtig for projektet
Udførelse	Max 1 min
Hyppighed	Af og til.
Åbne spørgsmål	Web applikationen vil i dette projekt kun være en simpel test version, så selve måden hvorpå man vælger parametere skal måske udbygges senere hen.
Afslutnings dato	31. August 2007

System sekvens diagram



Figur 10 – System sekvens diagram UC 4

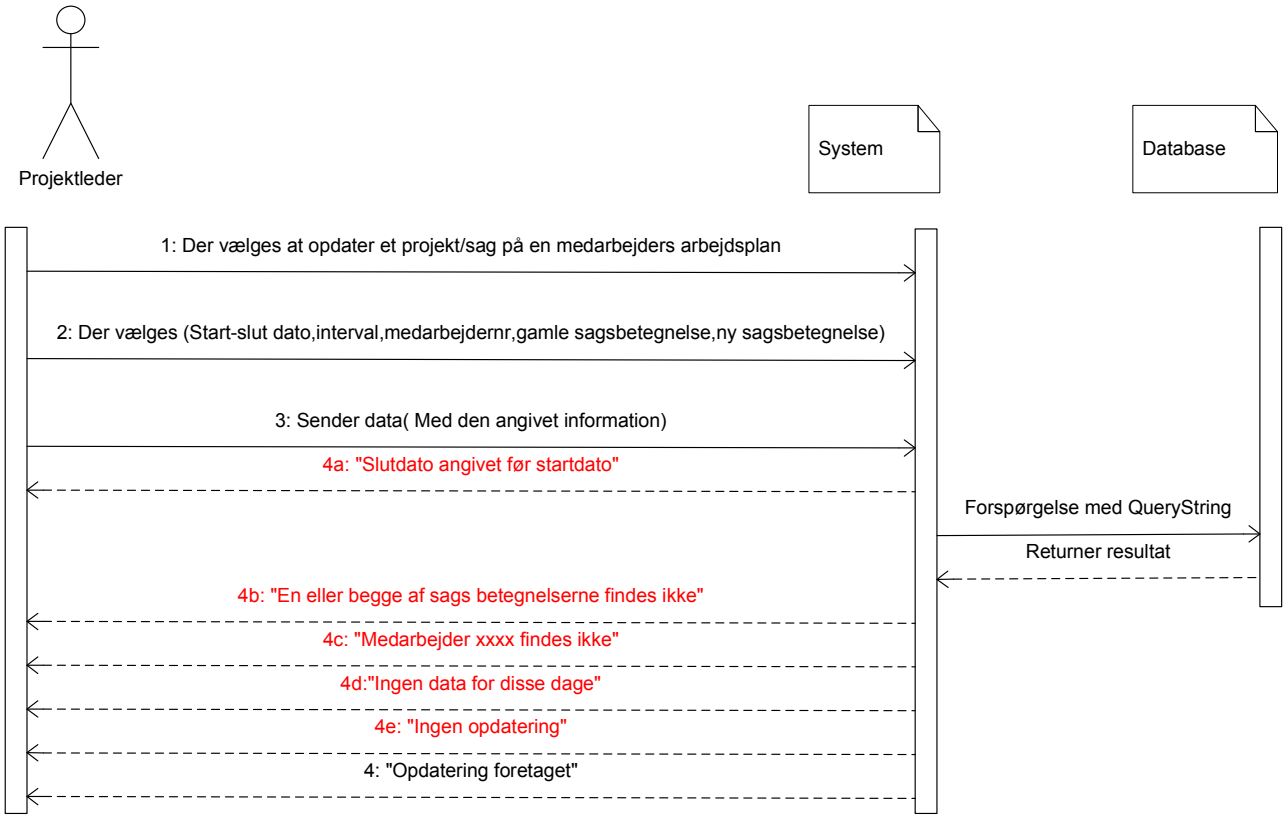
4.2.4 UC 14 Opdater en sag eller projekt på en medarbejders arbejdsplan

Use case ID	UC 14	**Use case navn**	Opdater en sag eller projekt på en medarbejders arbejdsplan.
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	**24. August 2007**	**Sidste opdaterings dato**	**29. August**
Beskrivelse af mål med use case	Målet med denne use case er, at give projektlederen mulighed for at opdatere en medarbejders arbejdsplan. Dvs. mulighed for at ændre en allerede tildelt sag/projekt til en anden for en given periode.		
Forudsætninger	UC 6 og UC 7 er blevet kørt først. Hvor ved alle medarbejders initialer findes for en specifik afdeling, og alle sagsbetegnelserne for projekter og sager findes.		
Succes kondition	Opdatering blev foretaget med hensyn til at udskifte den gamle tildeling ud med den nye for den angivne periode..		
Fejl kondition	Opdatering kunne ikke foretages.		
Primær Aktør	Projektleder.		
Trigger	Der ønskes at opdatere et tildelt projekt/sag på en medarbejders arbejdsplan for en given periode til en anden..		
Normalt forløb	1**.** Brugeren vælger at opdatere en sag eller projekt på en medarbejders arbejdsplan til en anden. 2. Brugeren vælger følgende for at opdatere en medarbejderens arbejdsplan: • En periode med start og slut dato. • Et interval(Formiddag, Eftermiddag eller hele dagen). • Medarbejder initialer. • En sagsbetegnelse for det som skal ændres i den angivne periode. • En sagsbetegnelse for det som skal tildeles i steder for. 3. Brugeren vælger at opdatere ud fra de valgte oplysninger. 4. System returnerer en meddelelse "Opdatering foretaget", hvis opdatering kunne gennemføres for medarbejderens arbejdsplan.		
Alternativt	**Step**	**Branching Action**	

forløb		
	4a	System returnerer fejlmeddelelsen ”Slutdato angivet før startdato”, hvis slutdato er angivet før startdato.
	4b	System returnerer fejlmeddelelsen ”En eller begge sagsbetegnelserne findes ikke”, hvis sagsbetegelserne ikke findes i systemet.
	4c	System returnerer fejlmeddelelsen ”Medarbejder xxxx findes ikke”, hvis medarbejder ikke findes i systemet.
	4d	System returnerer fejlmeddelelsen ”Ingen data for disse dage”, hvis der ikke findes noget at kunne opdatere på for perioden.
	4e	System returnerer fejlmeddelelsen ”Ingen opdatering”, hvis noget går galt under opdatering i databasen.

Beslægtet information	For at se om det er muligt at opdatere noget på en medarbejders arbejdsplan i perioden henføres til use case UC2.
Prioritet	48 ud af 100 vigtig for projektet
Udførelse	Max 1 min
Hyppighed	Af og til.
Åbne spørgsmål	Webapplikationen vil i dette projekt kun være en simpel test version, så selve måden hvorpå man vælger parametere skal måske udbygges senere hen.
Afslutnings dato	31. August

System sekvens diagram



Figur 11 – System sekvens diagram UC 14

4.2.5 UC 9 Vis arbejdsbelastning for medarbejder

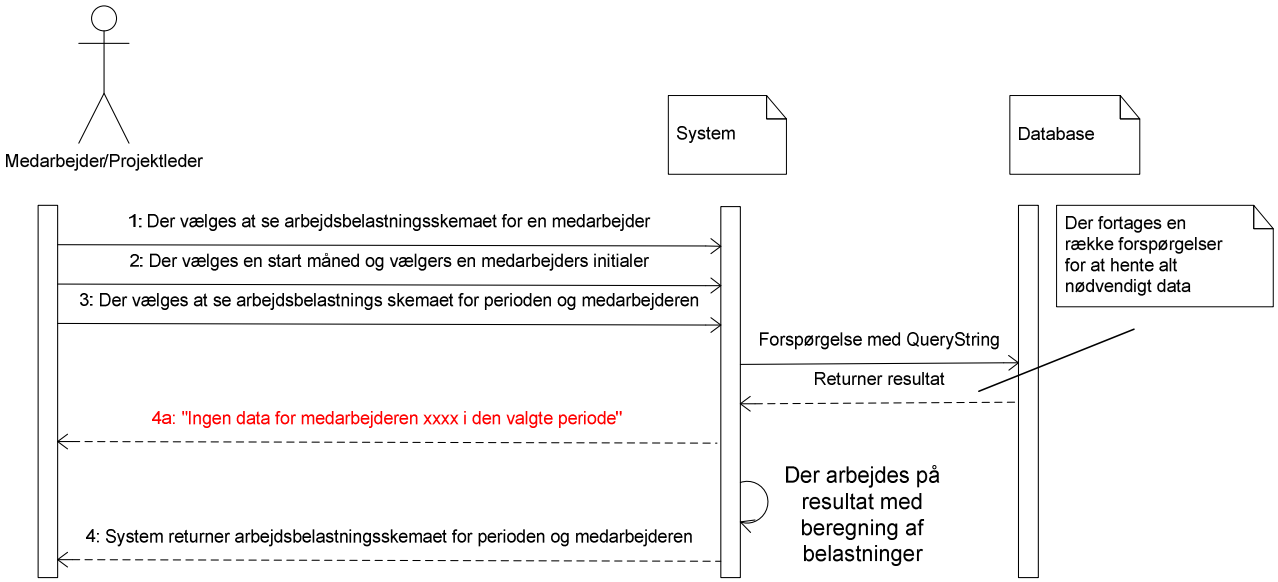
Use case ID	UC 9	**Use case navn**	Vis arbejdsbelastning for medarbejder
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	7. september 2007
Beskrivelse af mål med use case	Målet med denne use case er at vise et arbejdsbelastningsskemaet for en medarbejder 1 år frem i tiden ud fra en valgt startmåned. • Man skal for hver måned se arbejdstimerne på de enkelte projekter/sager delt op ud fra de forskellige typer. • Der skal for hver måned beregnes en månedens belastning i % ud fra timer på projekter/sager i forhold til månedens norm arbejdstimer.. • Månedens forventede - og sikre uddebiterings grad skal også		

	beregnes med hensyn til de projekter/sager, som kan faktureres ud til de kunder, som medarbejderen er tildelt, så man herved kan se, hvordan det vil gå med hensyn til indtjening for denne medarbejder frem i tiden.	
Forudsætninger	UC 6 køres hvorved alle medarbejdernes initialer findes for en specifik afdeling.(På dette tidspunkt GIS & IT).	
Succes kondition	Arbejdsbelastningsskemaet for den valgte medarbejder og periode bliver vist korrekt.	
Fejl kondition	Arbejdsbelastningsskemaet kunne ikke laves for den valgte medarbejder og den valgte periode.	
Primær Aktør	Projektleder.	
Sekundær Aktør	Medarbejder	
Trigger	En medarbejder eller projektleder ønsker at se arbejdsbelastningsskemaet for en medarbejder.	
Normalt forløb	1. Der vælges at se arbejdsbelastningsskemaet for en medarbejder. 2. Der vælges en start måned og vælges en medarbejder med initialer. 3. Der vælges at se arbejdsbelastningsskemaet for den valgte periode og den pågældende medarbejder. 4. System returnerer arbejdsbelastningsskemaet for den valgte medarbejder og periode (1 år) med alle de nødvendige data.	
Alternativt forløb	**Step**	**Branching Action**
	4a	System returnerer en fejlmeddelelse ”Ingen data for medarbejderen xxxx i den valgte periode”, hvis arbejdsbelastningen ikke kan laves enten fordi ingen dato er valgt, eller fordi medarbejderen ikke findes i systemet.

Beslægtet information	Data i det returnerede skema vil indeholde de fleste af elementerne fra det nuværende Excel skema, som viser arbejdsbelastningen.
Prioritet	48 ud af 100 vigtig for projektet
Udførelse	Visning skal foretages hurtigt indenfor max 1 min
Hyppighed	Dagligt

Åbne spørgsmål	Webapplikationen vil i dette projekt kun være en simpel test version, så selve måden hvorpå man vælger parametere skal måske udbygges senere hen.
Afslutningsdato	7. september 2007

System sekvens diagram



Figur 12 – System sekvens diagram UC 9

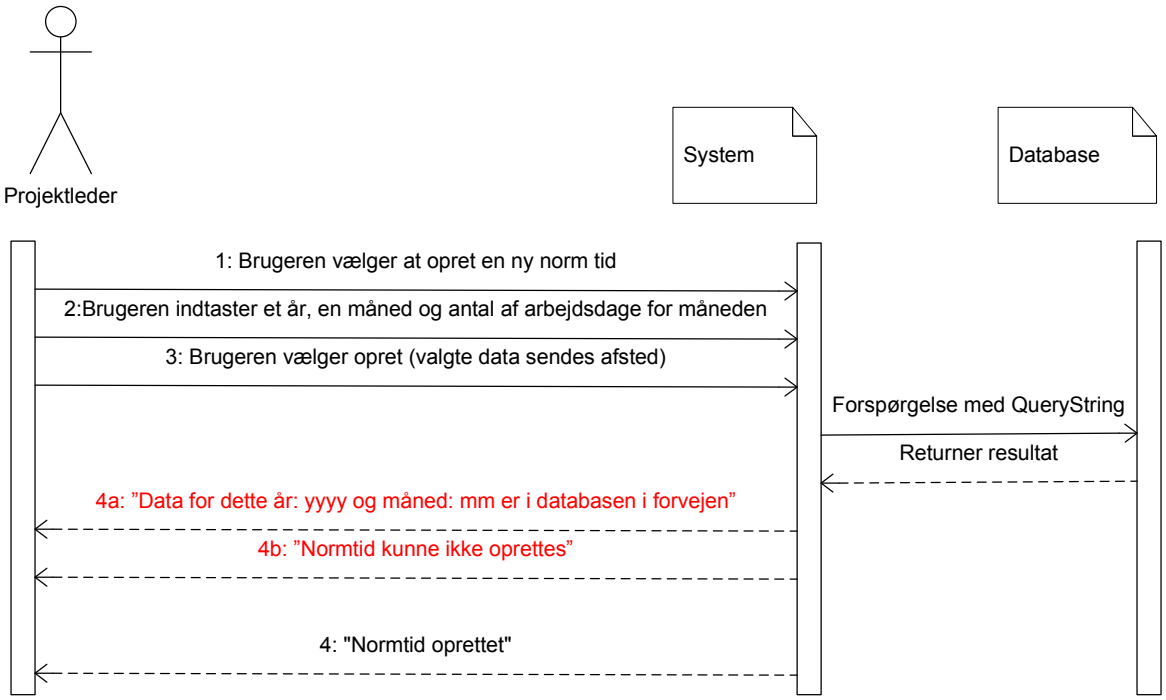
4.2.6 UC 15 Opret normtider

Use case ID	UC 15	**Use case navn**	Opret normtider
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	7. september 2007
Beskrivelse af mål med use case	Målet med denne use case er at give muligheden for at indtaste data vedrørende arbejdsdage for en bestemt måned til brug for beregning af arbejdsbelastning for en medarbejder.		
Forudsætninger			

Succes kondition	En ny norm tid bliver oprettet	
Fejl kondition	Norm tid kunne ikke oprettes	
Primær Aktør	Projektleder.	
Trigger	En medarbejder eller projektleder ønsker at se arbejdsbelastningsskemaet for en medarbejder.	
Normalt forløb	1. Brugeren vælger at oprette en ny normtid. 2. Brugeren angiver et år, en måned og antallet af arbejdsdage for måneden. 3. Brugeren vælger opret. 4. System returnerer meddelelsen "Normtid oprettet", hvis oprettelsen lykkes.	
Alternativt forløb	**Step**	**Branching Action**
	4a	System returnerer fejlmeddelelsen ”Data for dette år: yyyy og måned: mm er i databasen i forvejen”, hvis der allerede findes en rekord for dette år og måned.
	4b	System returnerer fejlmeddelelsen ”Normtid kunne ikke oprettes”, hvis data ikke kunne indsættes i databasen.

Beslægtet information	
Prioritet	10 ud af 100 ikke så vigtig
Udførelse	Oprettelse skal foretages hurtigt indenfor max 1 min
Hyppighed	Et par gange om året.
Åbne spørgsmål	Opret normtid vil ikke indgå i Webapplikationen i dette projekt, men vil indgå i web servicen, som en metode. Skal måske indføres i webapplikationen i arbejdet efter dette projekt.
Afslutningsdato	7. september 2007

System sekvens diagram



Figur 13 - System sekvens diagram UC 15

4.2.7 UC 10 Opret projekt/Sag

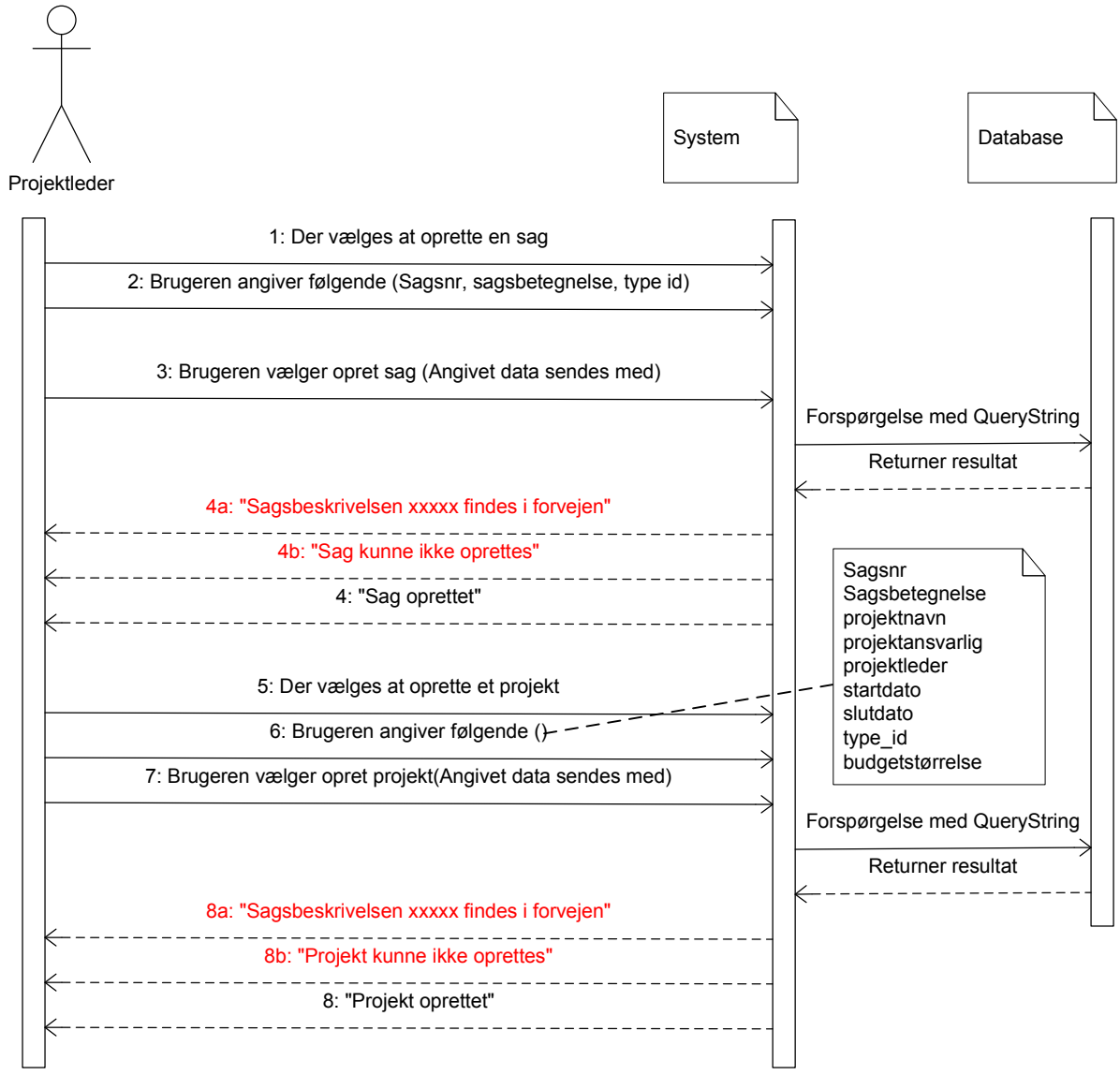
Use case ID	UC 10	**Use case navn**	Opret projekt/Sag
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	13. August 2007	**Sidste opdaterings dato**	10. september 2007
Beskrivelse af mål med use case	Mål med denne use case er at få oprettet et projekt eller en sag. Hvis der vælges at oprette en sag angives følgende data (Sagsnummer, Sags betegnelse, type id). Hvis der vælges at oprette et projekt angives følgende (Sags nummer, Sagsbetegnelse, projektnavn, projektansvarlig, projektleder, startdato, slutdato, type id og budgetstørrelse på projektet)		
Forudsætninger			
Succes kondition	Det lykkes at oprette en sag eller et projekt.		
Fejl kondition	Det kunne ikke lade sig gøre at oprette et projekt eller en sag.		
Primær Aktør	Projektleder		

Trigger	Der ønskes at oprette en ny sag eller et nyt projekt	
Normalt forløb **For opret sag**	1. Brugeren vælger at oprette en sag 2. Brugeren angiver følgende: • Et sagsnummer • En sagsbetegnelse • Et type id for sagen 3. Brugeren vælger opret sag. 4. Systemet returnerer meddelelsen ”Sag oprettet”, hvis sagen kunne oprettes	
Normalt forløb **For opret projekt**	5. Brugeren vælger at oprette et projekt 6. Brugeren angiver følgende: • Et sags nummer • En sagsbetegnelse • Et projekt navn • En projektansvarlig • En projektleder • En start- og en slutdato • Et type id for projektet • Budgetstørrelsen for projektet 7. Brugeren vælger opret projekt 8. Systemet returnerer meddelelsen ”Projekt oprettet”, hvis projektet kunne oprettes.	
Alternativt forløb	**Step**	**Branching Action**
	5a	Systemet returnerer fejlmeddelelsen ”Sagsbeskrivelsen xxxxx findes i forvejen”, hvis sagsbetegnelsen findes i forvejen.
	5b	System returnerer fejlmeddelelsen ”Sag kunne ikke oprettes**”,** hvis noget går galt under indsættelse i databasen.
	8a	System returnerer fejlmeddelelsen ”Sagsbeskrivelsen xxxx findes i forvejen”, hvis sagsbetegnelsen findes i forvejen.
	8b	System returnerer fejlmeddelelsen ”Projekt kunne ikke oprettes”, hvis noget går galt under indsættelsen i databasen.

Beslægtet	

information	
Prioritet:	10 ud af 100 vigtig men ikke så kritisk.
Udførelse	Det skal max tage et 1 min at oprette en sag eller et projekt
Hyppighed	Når nye sager eller projekter oprettes.
Åbne spørgsmål	Oprettelsen af sager og projekter i systemet skal måske i fremtiden, når Atkins får et nyt økonomisystem, gøres ved udtrækning fra dette.
Afslutnings dato	14. september 2007

System sekvens diagram



Figur 14 - System sekvens diagram UC 10

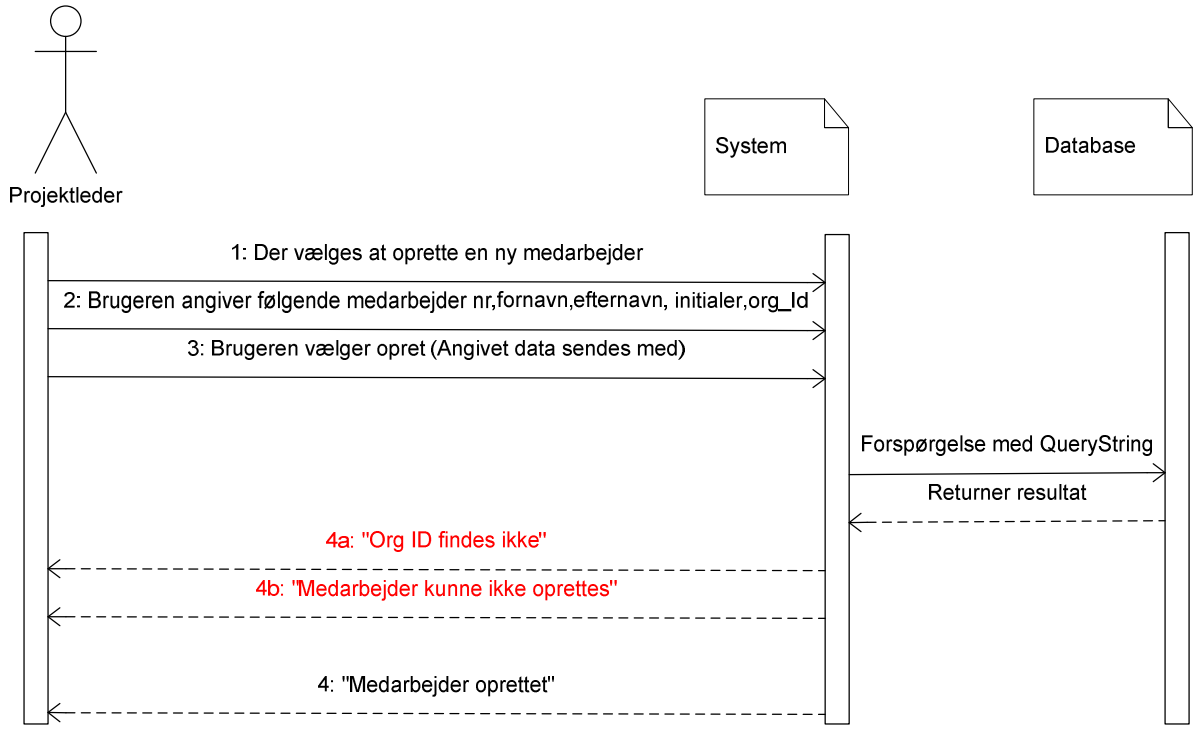
4.2.8 UC 11 Opret medarbejder

Use case ID	UC 11	**Use case navn**	Opret medarbejder
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	10. september 2007
Beskrivelse af mål med use case	Mål med denne use case er at få oprettet en medarbejder i systemet med følgende data (medarbejdernummer, fornavn, efternavn, initialer), samt at få denne tilknyttet i organisationen ved angivelse af et organisations id. Dette id fortæller hvilken organisation, sektor, afdeling og evt. underafdeling man tilhører.		
Forudsætninger			
Succes kondition	Den nye medarbejder bliver oprettet korrekt i systemet.		
Fejl kondition	Medarbejderen kunne ikke oprettes i systemet		
Primær aktør	Projektleder		
Trigger	Der skal oprettes en ny medarbejder i systemet..		
Normalt forløb	1. Brugeren vælger at oprette en ny medarbejder. 2. Brugeren angiver følende • Et medarbejdernummer. • Fornavn • Efternavn • Initialer. • Et organisations id 3. Brugeren vælger opret 4. System returnerer meddelelsen ”Medarbejder oprettet”, hvis medarbejderen kunne oprettes.		
Alternativt forløb	**Step**	**Branching Action**	
	4a	System returnerer fejlmeddelelsen ”Org ID findes ikke”, hvis der ikke der ikke findes nogen form for organisation med	
	4b	System returnerer fejlmeddelelsen ”Medarbejder kunne ikke oprettes”, hvis medarbejder ikke kunne oprettes. Dette kan skyldes at en medarbejder med det angivet medarbejdernummer findes i	

		forvejen.

Beslægtet information	
Prioritet:	10 ud af 100 vigtig men ikke så kritisk
Udførelse	Oprettelse skal max tage 1 min
Hyppighed	Hver gang en ny medarbejder bliver ansat
Åbne spørgsmål	Oprettelsen af medarbejdere i systemet skal måske i fremtiden, når Atkins får et nyt økonomisystem, gøres ved udtrækning fra dette.
Afslutnings dato	14. september 2007

System sekvens diagram

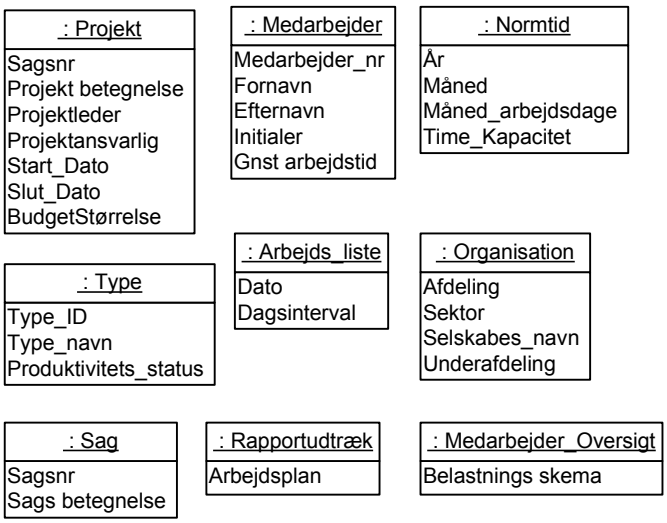


Figur 15 - System sekvens diagram UC 11

4.3 Domain model

4.3.1 Identifikation af begrebsmæssige klasser og attributter

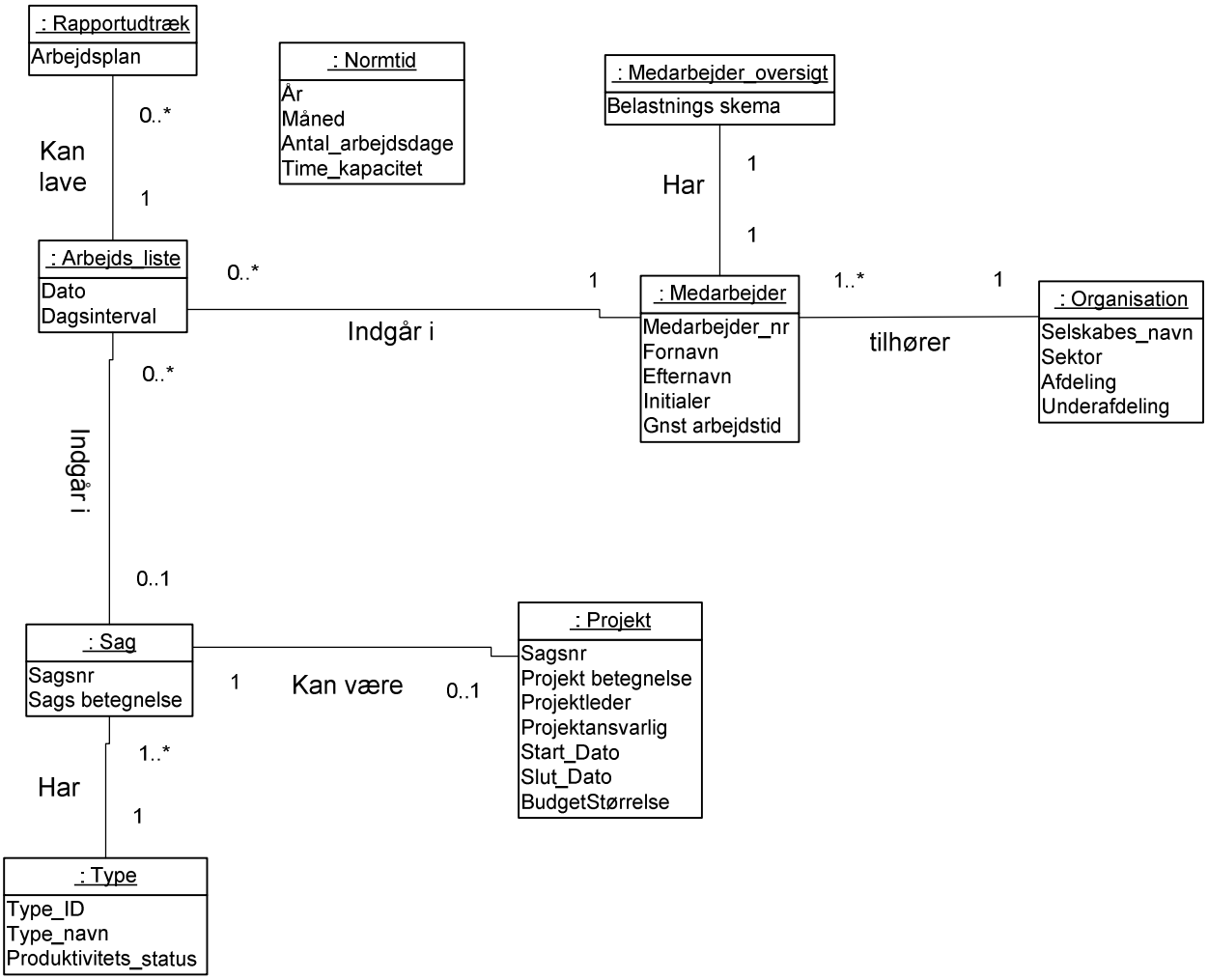
For at danne mig en bedre forståelse af systemet har jeg identificeret begrebsmæssige klasser og egenskaber ved at gennemgår mine use cases for navneord og lavet en brainstorm på projektet. På baggrund af dette har jeg fundet følgende begrebsmæssige klasser og egenskaber.



Figur 16 begrebsmæssige klasser og egenskaber.

4.3.3 Domain model diagram

En domain model skabes ud fra de fundne begrebsmæssige klasser og egenskaber ved at lave forbindelser mellem de klasser, som har relationer mellem hinanden. Herved skabes en beskrivelse af systemet på et højt abstraktions niveau. Det skal dog lige siges, at domain modellen ikke har noget med software klasser at gøre, men kan bruges til at få en ide over de klasser og de egenskaber, som klasserne skal indeholde under implementering af systemet.



Figur 17 Domain model over konceptet for systemet.

Kort fortalt indeholder domain modellen en begrebsmæssig klasse medarbejder, som har en række nødvendige oplysninger, denne skal bruge i et planlægningssystem. En medarbejder tilhører en eller anden form for organisation, f.eks. kunne det være selskabet Atkins, sektoren Miljø & GIS og afdeling GIS & IT. En medarbejder har også en oversigt, hvor denne kan se sit arbejdsbelastnings skema. En medarbejder indgår også i en arbejdsliste, som kan holde styr på, hvad den enkelte medarbejder arbejder med på en bestemt dag og interval(Formiddag/eftermiddag) på denne dag. Fra arbejdslisten kan man lave et eller flere rapportudtræk og få lavet en arbejdsplan.

I arbejdslisten indgår en række sager. En sag indeholder data om denne og kunne f.eks. være ferie, barsel. En sag kan dog også være en del af et projekt. Et projekt indeholder relevante oplysninger om dette, såsom en start og slutdato, samt en budgetstørrelse.
Hver enkelt sag har en type, som siger noget om sagen, f.eks. om den er intern og ikke produktiv, eller om det er en igangværende sag, som er produktiv, dvs. man kan fakturere en regning ud til kunden. Desuden er en normtid noget, som er vigtigt i systemet, da denne kan give oplysninger om en bestemt måneds antal normale arbejdsdage og arbejdstimerne i denne måned.

4.4 Resume

I analysen fik jeg givet en ide til, hvordan systemet skal opbygges. Samtidig fik jeg lavet en detaljeret beskrivelse og et sekvens diagram for de valgte use case under hver iteration. Til sidst fik jeg lavet en domain model, som giver en begrebsmæssig beskrivelse af systemet på et højt abstraktionsniveau. Denne vil senere under design og implementation af de enkelte use cases kunne giver mig ideer til klasser og egenskaber.
Under mit arbejde i de første iteration har jeg fundet frem til at UC 12, UC 13 og UC 3 [se punkt 3.3.3] udgår af dette projekt eftersom UC 12 og UC 13 på nuværende tidspunkt kun er en ekstra feature, da de allerede findes i et andet system. UC 3 med visning af frie ressourcer kan allerede ses under arbejdsplanen for medarbejderne, derfor udgår denne også.

Kapitel 5

Design

5. Design

5.1 Indledning fra analysen til design

I design afsnittet går jeg fra det højere abstraktions niveau med beskrivelser af brugerens brug af systemet(use cases) og domain modellen med begrebsmæssige klasser og egenskaber til det reelle design af software klasser og egenskaber til realisering af use casene i systemet. Denne procces kaldes use case realisering.
Til dette benytter jeg klasse/sekvens diagrammer, som laves samhørende under design af de enkelte use case. Klassediagrammet viser, hvilke egenskaber og metoder de enkelte klasser indeholder. Samtidig viser det også, hvilken associationer der er imellem de enkelte klasser.
Sekvensdiagrammet viser software forløbet mellem klasserne ned igennem systemet for de enkelte use case. Softwareklasserne identificeres vha. af domain model fra analyse afsnittet. Det skal dog siges, at ikke alle de begrebsmæssige klasser og egenskaber bliver til softwareklasser eller egenskaber, samtidig kan der også laves flere klasser og egenskaber end der er af begrebsmæssige klasser og egenskaber.

5.2 Design patterns

Til dette projekt har jeg valgt at benytte 2 design patterns, som jeg synes giver mening for dette projekt.
Det første er brugen af et kodebibliotek fra Microsoft kaldet DAAB (data access aplication block), som er et pattern, som gør det lettere at lave data tilgangen til database del i mit projekt.
Det andet, jeg har valgt, er et singleton pattern, som sørger for, at kun en instans af en klasse kan oprettes.

5.2.1 DAAB

Enterprise Library er en samling af applikations blokke, som er udgivet af Patterns og Pratices gruppen indenfor Microsoft. Der er 7 blokke man kan gøre brug af, som hver især bygger på bedste praksis indenfor hvert deres område.
Jeg vil til mit projekt gøre brug af ”*Data Access*” blokken (DAAB) under Microsoft Enterprise Library January 2006. DAAB gør at jeg sparer at skulle skrive en del linier kode med hensyn til mine databasekald.

```
// Connect to the database
SqlConnection myConnection = new SqlConnection(connection string);
myConnection.Open();

// Specify the command to use to query the database
SqlCommand myCommand = new SqlCommand(sql query, myConnection);

// If the SQL query has parameters (i.e., @ParamName), add values for
// the parameters in the query...
myCommand.Parameters.Add("@ParamName1", ParamValue1);
myCommand.Parameters.Add("@ParamName2", ParamValue2);
...
myCommand.Parameters.Add("@ParamNameN", ParamValueN);

// Retrieve the results of the query in a SqlDataReader
SqlDataReader reader = myCommand.ExecuteReader();

// work with the returned data...
```

Figur 18 eksempel med brug af ADO.NET klasserne

I overnævnte eksempel skal man skrive en del linier kode bare for at få hentet data fra en database, hvilket gør, at der er en større sandsynlighed for at begå fejl. Dette prøver DAAB at gøre bedre ved at reducere mængden af kode, man skal skrive.

```
 // Create a database object
Database db = DatabaseFactory.CreateDatabase();

// Get back a DataReader
IDataReader reader = db.ExecuteReader(CommandType.Text, SQL query);
```

Figur 19 eksempel med brug af DAAB

I overnævnte eksempel er koden reduceret meget i forhold til figuren 18 for udførelses af det samme.

DAAB henter ved kørsel data provider informationen fra config.web filen.

```
<configSections>
<section name="dataConfiguration"
type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
Microsoft.Practices.EnterpriseLibrary.Data, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=null" />
</configSections>
<dataConfiguration defaultDatabase="Planning" />
<connectionStrings>
    <add name="Planning"
          connectionString="Database=DB_Planning;Server=(local)\SQLEXPRESS;Inte
grated     Security=SSPI;User ID=Thomas;Password=password!;"
          providerName="System.Data.SqlClient" />
</connectionStrings>
```

Figur 20 eksempel på config.web filen

Herved understøtter DAAB brugen af en abstrakt provider, hvilket vil gøre det lettere på et senere tidspunkt at udskifte databasen til en anden, som er .net kompatibel.

5.2.2 Singleton

Singleton bruges i de tilfælde, hvor der kun er brug for en instans af en klasse. Alle objekter, der skal bruge en instans af klassen, vil bruge den samme instans. Singleton kan implementeres ved, at klassen har en private static instans af den selv, og en private Constructor, som sørger for, at den ikke kan bruges af andre klasser til at danne instanser af et objekt. For at andre skal kunne få en instans fra denne klasse laves en get egenskab, som returnerer instansen af klassen.

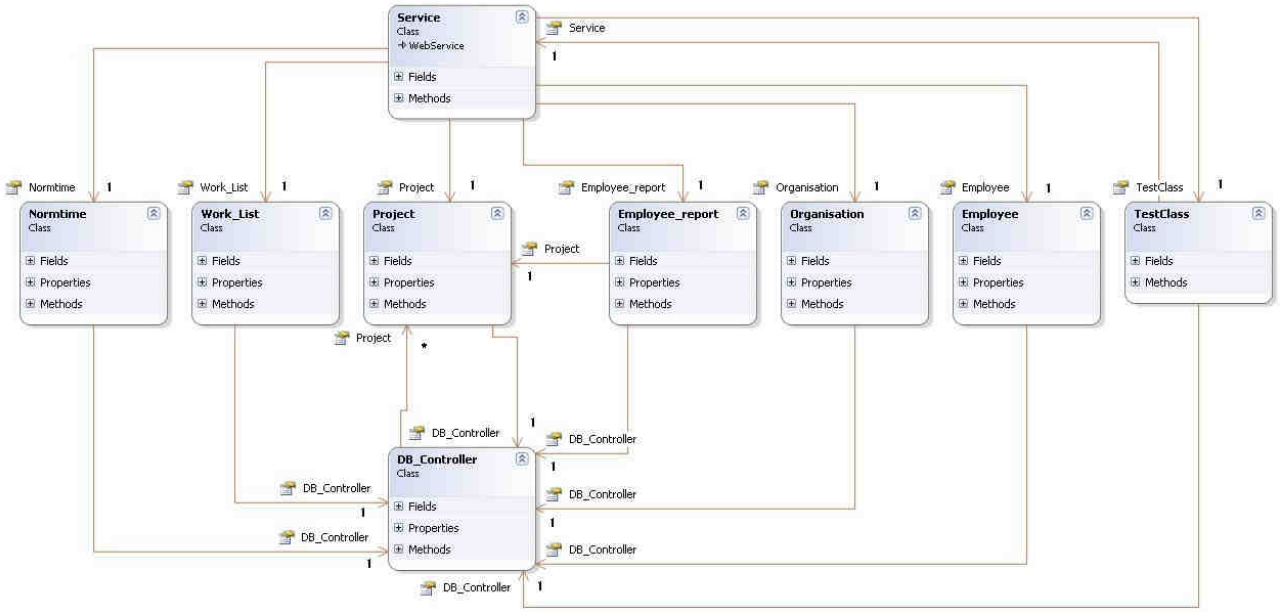
```
public class DB_Controller
{
     private static DB_Controller DBinstance = new DB_Controller();

     private DB_Controller(){}
     public static DB_Controller GetDBinstance
     {
         get{return DBinstance;}
     }
}
```

Figur 21 Eksempel på brugen af singleton pattern

Det kunne være klogt kun at have en instans af klassen, som styrer tilgangen til databasen, derfor vil jeg bruge singleton på den klasse, som jeg vil have til at styre tilgang til databasen.

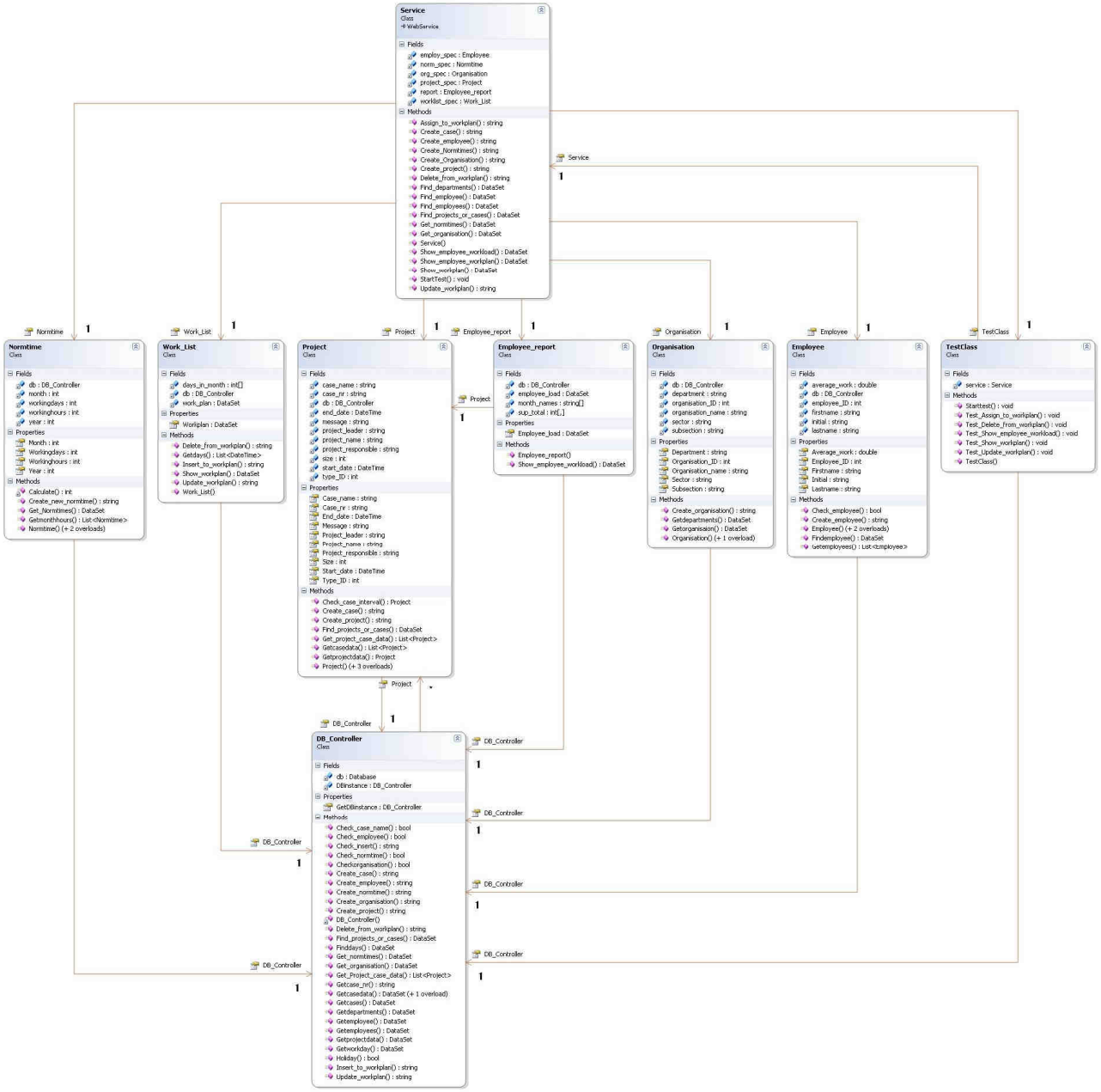
5.3 Klassediagram

Klassediagrammet er blevet lavet ud fra analysen med use case beskrivelserne og domain modellen. Har løbende lavet diagrammet med klasser og metoder efterhånden, som flere use cases er blevet lavet. For at få det endelige diagram tegnet har jeg brugt ”reverse engineering” på koden.



Figur 22 Klasse diagram for web service delen med visning af associationer

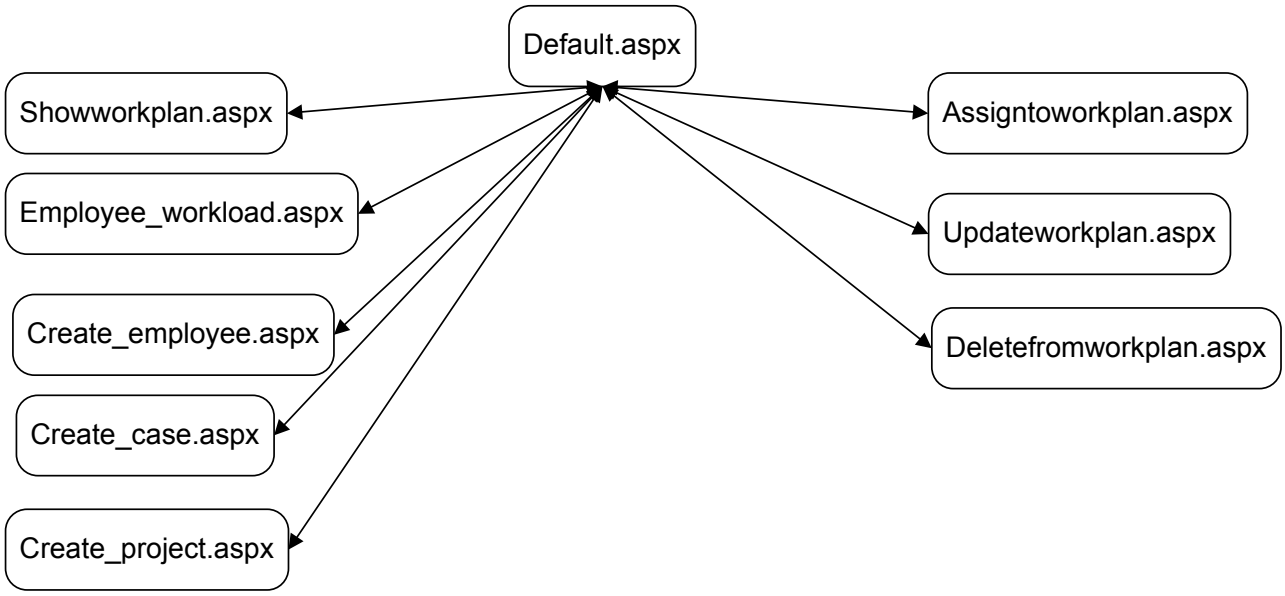
I forhold til domain modellen har jeg valgt at have alt med type, sager og projekter til at indgå i Project klassen, da de hører sammen. Desuden har jeg valgt at lade arbejdsplanen indgå i klassen Work_List, da det er denne klasse, der er beregnet til at have med arbejdsplanen at gøre. Samtidig er alt med Organisationen samlet i en klasse.
Associationerne mellem klasserne skal laves således, at alle de klasser, som gør brug af DB_Contoller klassen, kun kan få en og samme instans af denne klasse, da singleton skal overholdes. Jeg har prøvet at tilstræbe et design princip med lav kobling mellem klasserne(få associationer), så modellen bliver lettere at overskue ved at give Service klassen ansvaret for at hente data fra de enkelte klasser og sende det videre til de klasser, som skal bruge det. Det er tildels opnået, men der er stadig nogle få rettelser, som skal laves i Employee_report og DB_Controller klassen. Løsning på dette vil blive forklaret i afsnittet 8.3 Evaluering.



Figur 23 Klasse diagram for web service delen med egenskaber og metoder.

I mit klasse diagram har jeg valgt ikke at tage parametere med til metoderne. Disse vil kunne ses under implementation kapitlet, hvor klassernes metoder vil blive forklaret.

5.4 Web diagram



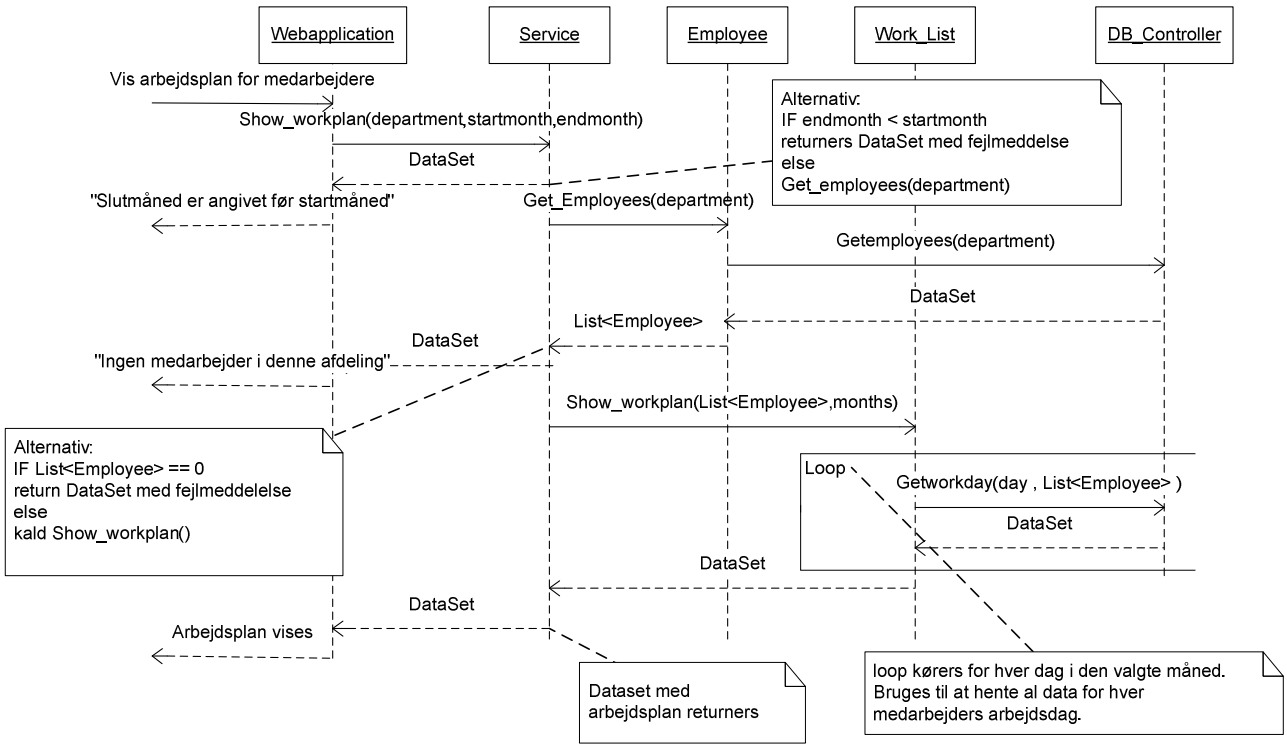
Figur 24 Web diagram med siderne til web delen, som bruges til at vise brugen af web servicen.

Har valgt at lave et simpelt web diagram med en række web sider. Siderne vil hver især omhandle nogle af de use case, som jeg har valgt at vise i webdelen. Dvs. ikke alle de implementeret use case vil være i webdelen. De vil til gengæld kunne bruges ved direkte brug af web servicen. Det er først meningen, at den rigtige webdel skal laves senere.

5.5 Sekvens diagrammer for use cases.

I analysen fik jeg lavet system sekvens diagrammer for de enkelte use cases, som viste brugerens interaktion med systemet, og ikke hvordan interaktionen var i selve systemet. Under designdelen er det derfor nødvendigt at lave nogle sekvens diagrammer for de enkelte use cases, som viser hvordan interaktion mellem software klasserne i systemet skal være. Derfor vil jeg i dette afsnit vise 3 af de vigtigste sekvens diagrammer og fortælle lidt om, hvad der sker. De andre sekvens diagrammer vil kunne ses i bilag D.

5.5.1 Sekvens diagram for UC2



Figur 25 Sekvens diagram for UC2

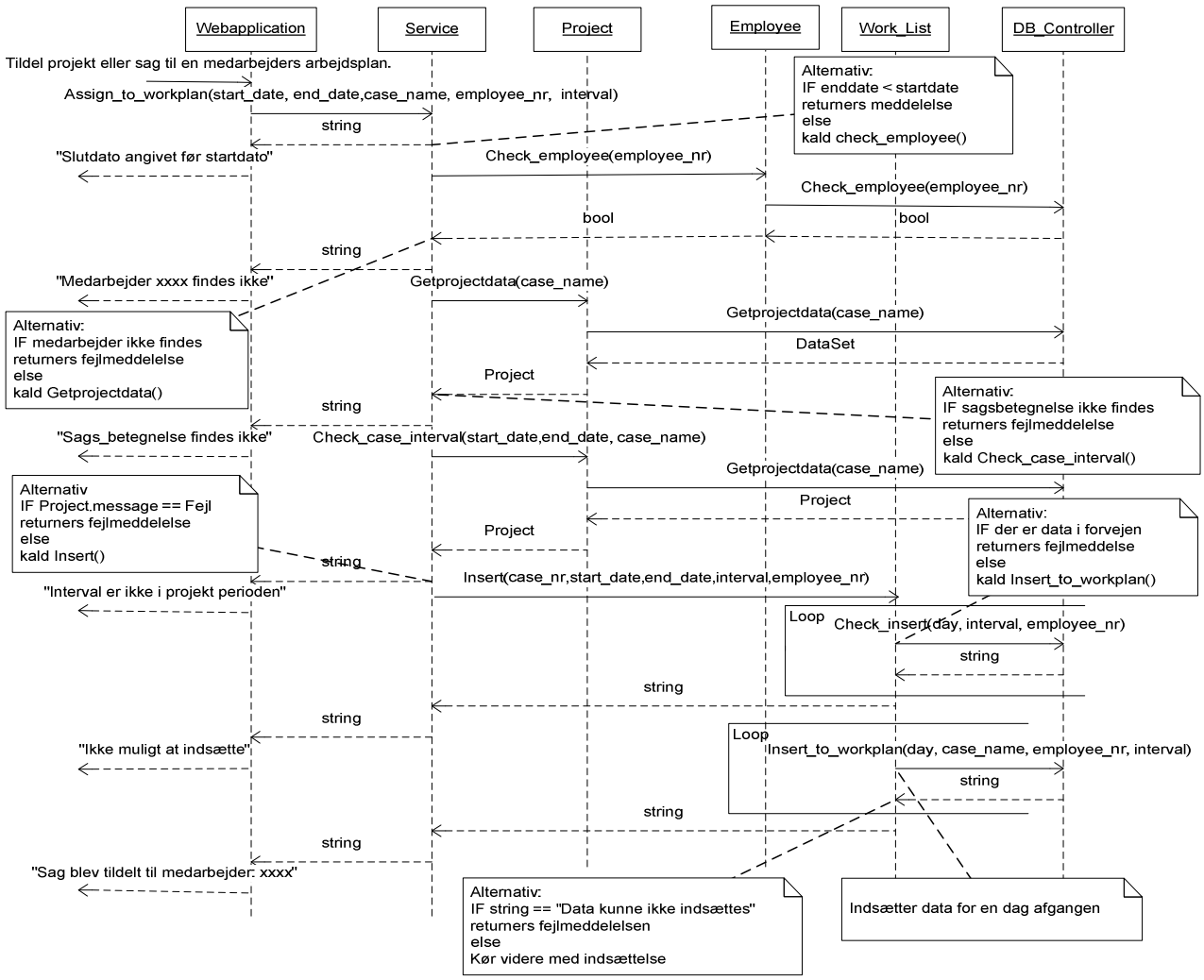
Figur 25 viser interaktion mellem software klasserne, når en bruger vælger at se arbejdsplanen for medarbejdere i en bestemt afdeling og periode.
Denne information sendes videre med kald af web Service klassens metode Show_workplan(), som først checker, om månederne er angivet i en forkert rækkefølge, hvis det er tilfældet returneres et DataSet med en fejlmeddelelse, ellers hentes alle medarbejder for den angivet afdeling.

Service klassens metode Show_workplan() kalder Get_Employees() metoden fra Employee klassen med afdelingen som parameter. Denne metode returnerer en liste med objekter for alle medarbejdere efter kald til DB_Controllerens metode Getemployees(), som udfører forspørgelsen til Databasen.

Ved tilbagevendelse til Service klassens metode checkes for, om der overhovedet er nogle medarbejdere i den pågældende afdeling, hvis ikke returneres en fejlmeddelelse om dette, ellers fortsættes.

Herefter kaldes Show_workplan() metoden fra Work_list klassen med en liste af medarbejdere og en liste med alle månederne for den valgte periode, som parameter. I denne metode laves arbejdsplan for den pågældende afdeling og periode.
Data til arbejdsplanen hentes med metoden Getworkday() i DB_Controller klassen ved en loopkørsel, som køres for hver dag i den valgte måned, hvor alt data for hver medarbejders arbejdsdag hentes. Efter arbejdsplanen er blevet lavet med alt det nødvendige data, returneres et DataSet med denne.
Service klassens metode returnerer dette DataSet med arbejdsplanen til webapplikationen, som viser indholdet for brugeren.

5.5.2 Sekvens diagram for UC5

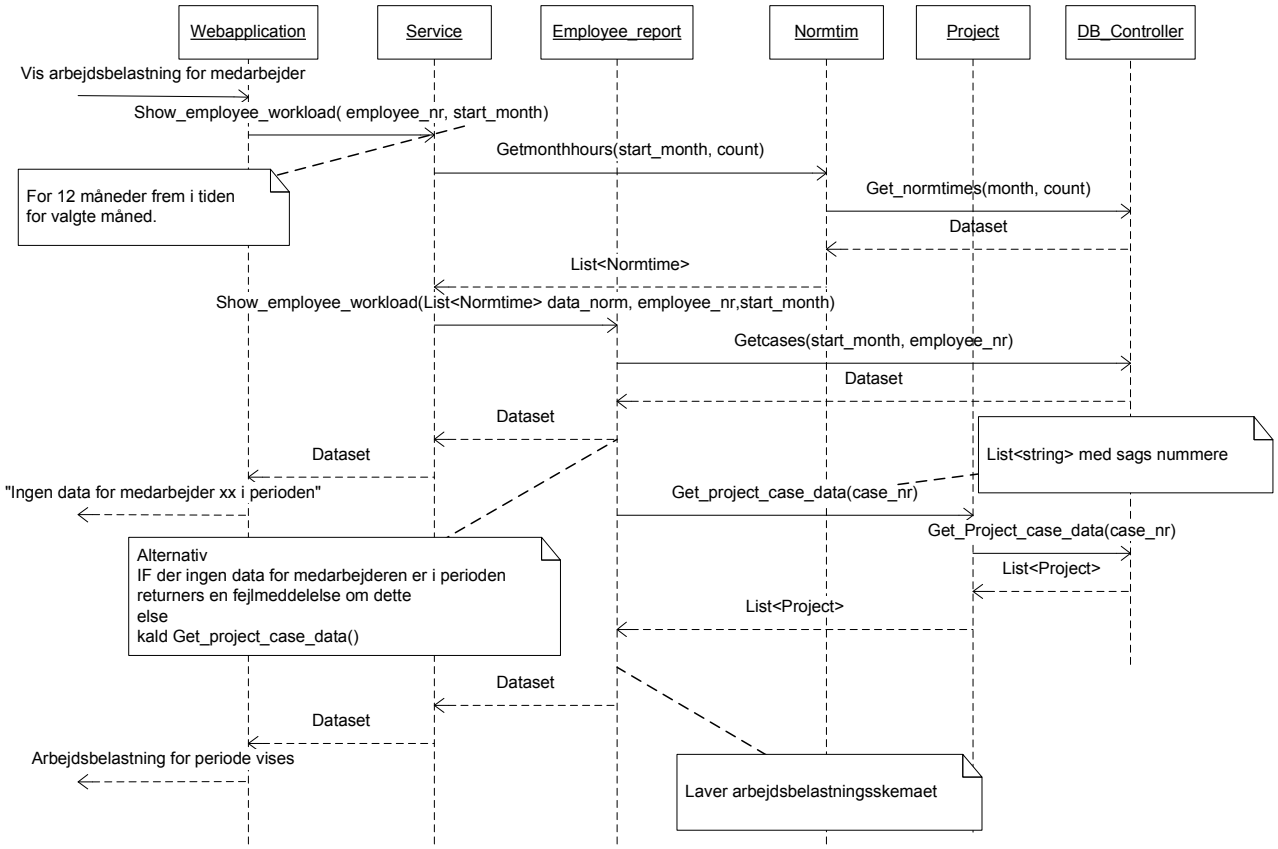


Figur 26 Sekvens diagram for UC5

Figur 26 viser interaktionen mellem klasserne, når en bruger vælger at tildele et projekt eller en sag til en medarbejders arbejdsplan og har angivet de nødvendige parametere.

Hovedtrækkende ved dette sekvens diagram er, at der skal udføres en del checks[se afsnit 4.2.2] på hentningen af de nødvendige data fra de repræsentative klasser inden en indsættelse af en sag/projekt i en medarbejders arbejdsplan kan ske. Hvis et check fejler, returneres en fejlmeddelelse.
De fleste af checkene sker i Service klassens metode Assign_to_workplan() inden metoden Insert() i Work_list klassen kaldes.
Insert metoden laver først et check på indsætningen i databasen, hvor der findes ud af, om sagen eller projektet kan indsættes i den valgte periode eller om perioden allerede er optaget af en anden sag.
Hvis indsættelsen er mulig, køres en loop, hvor metoden Insert_to_workplan() i DB_Controlleren kaldes for hver dag i den valgte periode med parametere, som skal bruges til indsættelsen, når indsættelsen er fuldført returneres en meddelelse med, at det lykkedes at indsætte sagen eller projektet i medarbejderens arbejdsplan.

5.5.3 Sekvens diagram for UC9



Figur 27 Sekvens diagram for UC9

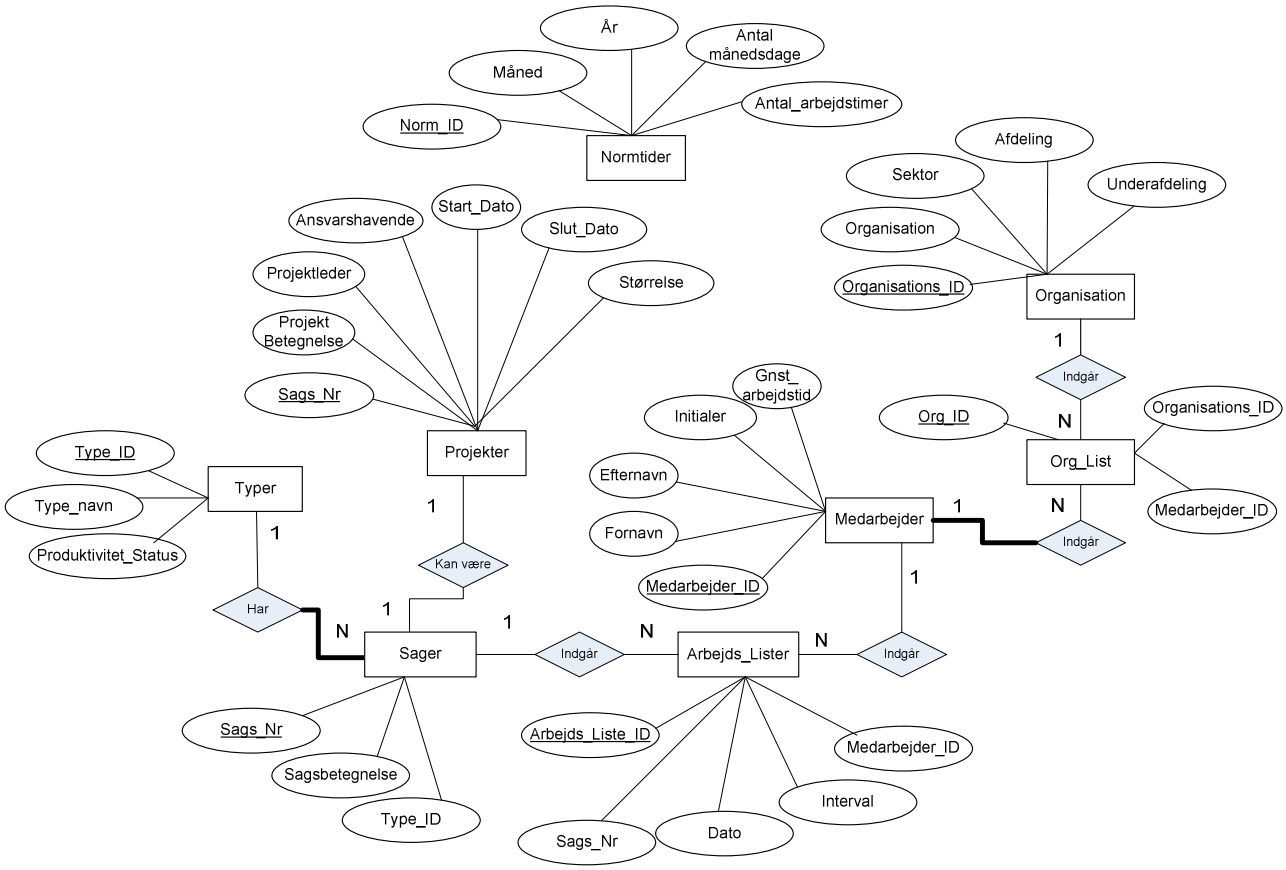
Figur 27 viser interaktionen mellem klasserne, når en bruger vælger at se en medarbejderes arbejdsbelastning og har angivet det nødvendige data i form af et medarbejdernummer og en startmåned.

Hovedtækkende ved dette sekvens diagram er Employee_Report klassens metode Show_employee_workload(), som har alt det nødvendigt data, som skal bruges til beregning af en medarbejders arbejdsbelastning, såsom en liste med normtider fra Normtime klassen for de 12 valgte måneder. Hvis der ikke findes nogle sagsnumre i medarbejderens arbejdsplan, returneres en fejlmeddelelse om dette. Hvis der er data for medarbejderen, hentes data for hvert enkelt sagsnr, og der oprettes en liste med 2 typer af objekter fra Project klassen, alt afhængigt af om det pågældende sagsnr er en sag eller også et projekt. Herefter laves arbejdsbelastningsskemaet for medarbejderen med de relevante beregninger ud fra det hentede data.

Når skemaet er lavet færdigt, returneres det i form af et DataSet, som vises i webapplikationen.

5.6 ER-Diagram for databasen

Ud fra de 2 tidligere systemer[se afsnit 1.1] og behov til en database struktur har jeg fundet frem til følgende entiteter og relationer mellem dem, samt hvilken attributter de enkelte entiteter skal kunne indeholde.



Figur 28 ER diagram

Entiteten medarbejder indeholder det data for en medarbejder, som er fundet nødvendigt fra de 2 foregående systemer, såsom et medarbejder_ID(nummer), navn, initialer og en attribut gennemsnits arbejdstid, som er beregnet til at indeholde oplysninger om arbejdstid for en medarbejder på uge basis.

En medarbejder skal indgå i mindst en Org_List, som sammenknytter, hvilken organisations sektor, afdeling og evt. underafdeling, medarbejderen tilhører ved at have en relation til organisation entiteten, som indeholder disse oplysninger. En Organisation kan indgår i mange Org_List. En medarbejder kan indgå i mange Arbejds_Lister, men behøver ikke at være det, eftersom en medarbejder ikke nødvendigvis er tildelt noget arbejde.

Arbejdslisten indeholder oplysninger om, hvilke sager medarbejderen arbejder med på bestemte datoer fordelt på intervallet formiddag/eftermiddag. Alle former for arbejde, som en medarbejder

kan tilknyttes i arbejds_listen, kan beskrives som en sag. Derfor kan sager indgår i flere arbejds_lister.
Alle former for arbejde kan beskrives vha. et sagsnr, en sagbetegnelse og et type_ID. Alle sager skal have en bestemt type. Der er 5 bestemte typer, som jeg har fundet frem til i de tidligere systemer. De kan beskrives vha. et id, navn og en produktivitet_status(True/false).

Typerne er følgende:

1. IGA, produktiv (Igangværende opgaver for kunder)
2. IGA, ikke produktiv (Interne opgaver, som ikke kan faktureres ud til en kunde)
3. Kom, produktiv (Kommende opgaver, som forventes at vindes)
4. Kom, ikke produktiv (Kommende opgaver fra tilbud og lignende, men som ikke kan forventes at vindes)
5. Div, ikke produktiv (Diverse opgaver f.eks. Ferie, barsel, orlov etc.)

En sag kan være et projekt, men behøver ikke at være det. Hvis en sag er et projekt bruges samme sagsnr, som i sager og alt relevant data for et projekt indgår i entiteten projekter, såsom størrelsen på et projekt pengemæssigt og start-slut dato for projektet.
Den sidste entitet er beregnet til at holde på normtidsdata for de enkelte måneder med hensyn til arbejdsdage og arbejdstimer i den pågældende måned. Entiteten har ikke nogen relationer til andre entiteter, eftersom den kun er tænkt som en entitet til at slå arbejdsdage og arbejdstimer op for bestemte måneder.
ER diagrammet viser, hvilke data jeg har fundet nødvendige til arbejdsplanlægningssystemet fra de 2 nuværende systemer og hvilke relationer, som findes mellem de forskellige entiteter.

5.7 Resume

I dette kapitel fik jeg på baggrund af analysen lave sekvensdiagrammer for de enkelte use cases og et klasse diagram over web service delen, samt et web diagram over test webdelen. Desuden fik jeg lavet et ER diagram over den mulige opbygning af databasen. Fik desuden også beskrevet hvilken 2 design patterns jeg havde tænkt mig at benytte ved implementation.

Kapitel 6

Implementation

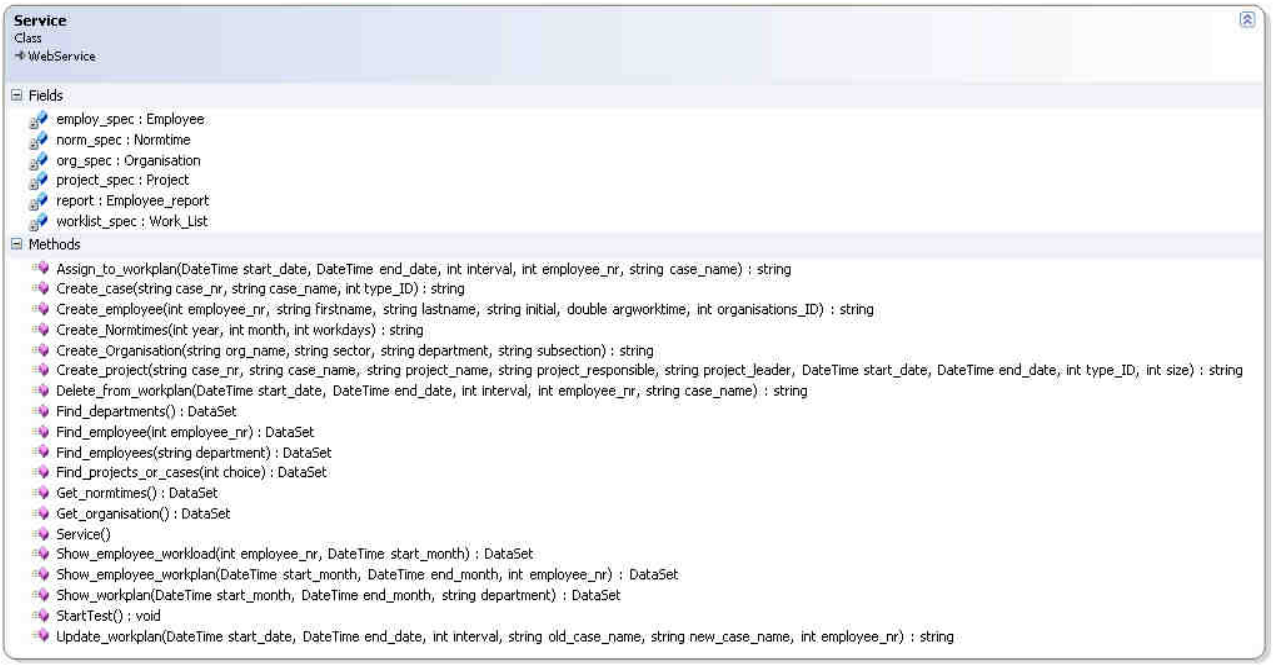
6. Implementation

6.1 Implementering af design

I dette afsnit har jeg tænkt mig at give en forklaring på nogle af de vigtigste metoder for de enkelte klasser i web service delen, som jeg har implementeret ud fra mit design.

6.1.1 Service klassen

Denne klasse er selve web service klassen, som har alle de metoder, som brugeren kan benytte sig af i sin webapplikation til arbejdsplanlægning.



Figur 29 Service klassen

Denne klasse har objekt referencer til alle de klasser, som den skal sende information videre til på nær DB_Controller klassen, da denne klasse ikke skal have noget med database del at gøre.

Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
`Show_workplan()`	DataSet	DateTime, DateTime, string.	Denne metode sørger for at hente al information for afdelingens medarbejdere og opretter en liste med alle månederne for periode, hvorefter denne information sendes videre til Work_List klassen, som returnerer arbejdsplanen i et DataSet. Dette DataSet returnerer metoden efterfølgende.
`Show_employee_workload`	DataSet	int, DateTime	Denne metode henter al normtidsdata, og sender denne information videre til Employee_report klassen, hvorfra den får et DataSet tilbage med medarbejderens belastningsskema.
`Create_project`	string	`string, string, string, string, string, DateTime, DateTime, int, int`	Denne metode sætter oprettelsen af et projekt i gang ved at sende information videre til Project klassen, som den får en meddelelse fra, om det er lykkes at oprette projektet. Denne besked returnerer metoden.

6.1.2 Employee_report klassen

Denne klasse har til formål at lave et belastningsskema for en medarbejder, derfor har den et DataSet employee_load som en member variabel, samtidig med at den har en instans af DB_Controller klassen ligesom de andre klasser, som den bruger

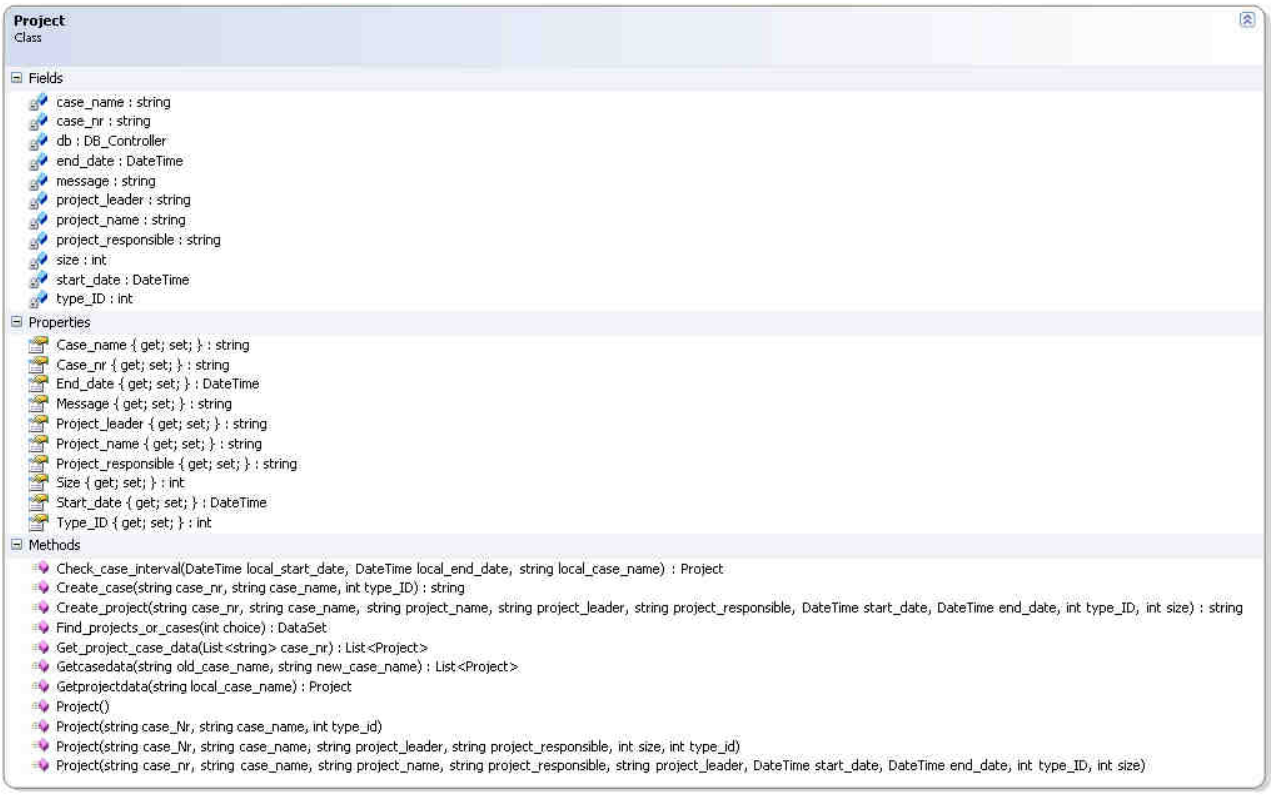
Figur 30 Employee_report klassen

Vigtige metoder:

Navn	**Retur type**	**Parameter**	**Beskrivelse**
Show_employee _workload	DataSet	List<Normtime>, int, DateTime.	Denne metode laver medarbejderens belastningsskema for 12 måneder frem. Fra startmåneden af med alle de relevante data, såsom hvor mange timer den enkelte medarbejder arbejder med hvert projekt pr måned, og hvordan time belastningen og uddebiteringsgraderne er pr måned.

6.1.3 Project klassen

Denne klasse repræsenterer projekt og sags objekter og har metoder til oprettelse og til at hente data for dem, samt relevante member variabler for disse.



Figur 31 Project klassen

Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
`Check_case_interval`	Project	`local_start_date, local_end_date, local_case_name`	Denne metode bruges under tildeling af projekt/sag til en medarbejders arbejdsplan. Den bruges til at checke, om en sag er et projekt. Hvis det er tilfældet, checkes på om en angivet periode ligger indenfor projektets kørselsperiode. Hvis ikke returneres et projekt objekt med en fejlmeddelelse.

6.1.4 Employee klassen

Denne klasse repræsenterer medarbejder objekter og har metoder til oprettelse og hentning af data af disse, samt relevante member variabler for disse.
Det skal dog siges, at på nuværende tidspunkt bruges average_work member variablen ikke, men kan tænkes at blive brugt på et senere tidspunkt som check ved tildelingen af sager til en medarbejder med nedsat arbejdstid.

Figur 32 Employee klassen

Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
`Getemployees`	List<Employee>	string	Denne metode bruges til at hente alt data for medarbejderne for en bestemt afdeling. Den får som input afdelingen og giver som output en liste med employee objekter.

6.1.5 Work_List klassen

Klassen bruges til at symbolisere arbejdsplanen for en medarbejder eller en afdelings medarbejdere. Den indeholder metoder til at hente arbejdsplan, samt indsætte, opdatere og slette sager/projekter på en medarbejders arbejdsplan.

Figur 33 Work_List klassen

Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
Getdays	List<DateTime>	DateTime, DateTime	Denne metode modtager en startdato og en slutdato, som den bruger til at lave en liste med DateTime objekter for hver dag i den pågældende periode. Metoden tager også forbehold for skudår.
Show_workplan	DataSet	List<Employee>, List<DateTime>	Denne metode modtager en liste med Employee objekter og en liste med måneder, som den bruger til at lave en arbejdsplan ud fra. Den returnerer arbejdsplan i form af et DataSet.
Insert_to_workplan	string	string, string, DateTime, DateTime, int, int	Denne metode modtager en række parametere, såsom sagsnr, start- og slutdato for indsættelse. Samt medarbejders nummer og hvilket

			interval(0 formiddag, 1 eftermiddag, 2 hele dagen) denne skal tildeles i perioden. Metoden returnerer en meddelelse om status på indsættelsen.

6.1.6 Normtime klassen

Denne klasse repræsenterer normtids objekter med deres member variabler. Bruges til at symbolisere en bestemt måneds arbejdsdage og arbejdstimer.

Figur 34 Normtime klassen

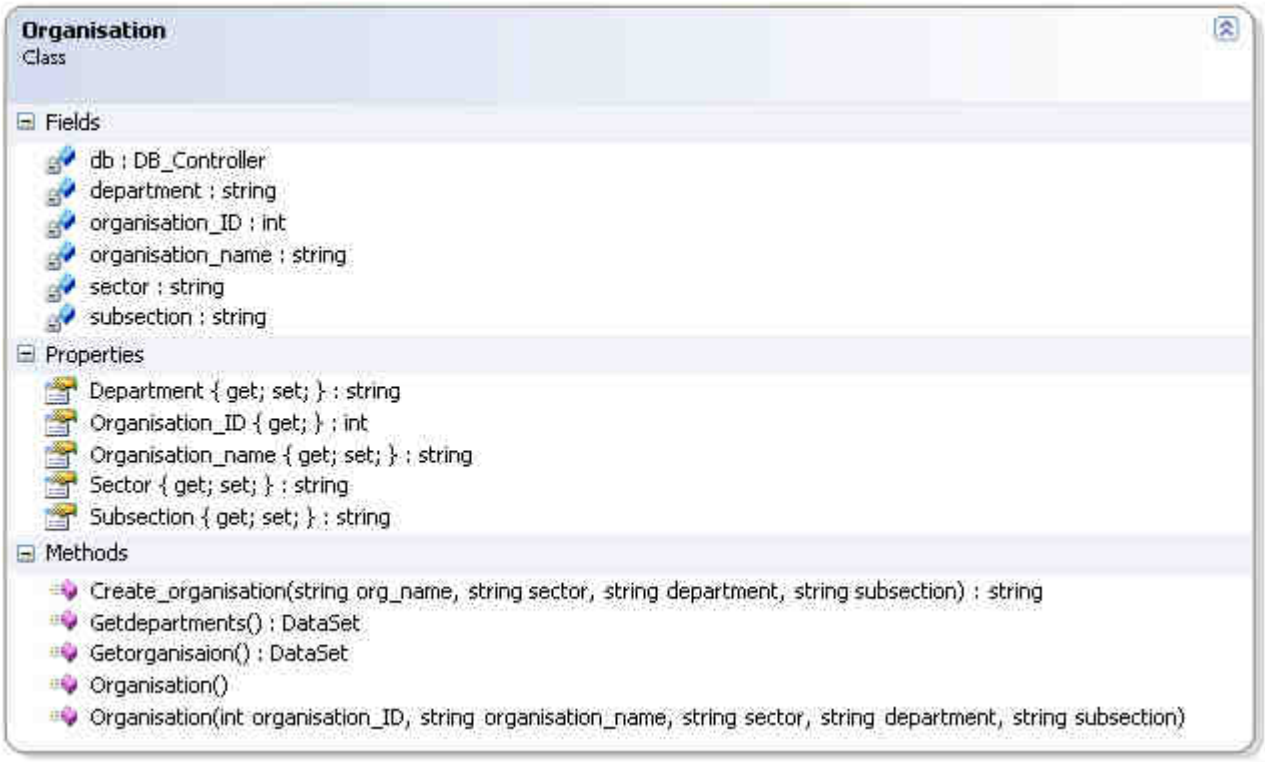
Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
Calculate	int	int	Denne metode får antallet af arbejdsdage i en måned ind og beregner antallet af arbejdstimer for denne måned, som bliver returneret.
Getmonthhours	List<Normtime>	DateTime, int	Denne metode får en måned ind, og

			antallet af måneder denne skal hente arbejdstimerne for. Den returnerer en liste med Normtime objekter, som er blevet initialiseret med antallet af arbejdstimer.

6.1.7 Organisation klassen

Denne klasse repræsenterer Organisations objekter med deres member variabler. Bruges til at symbolisere en bestemt organisations del.

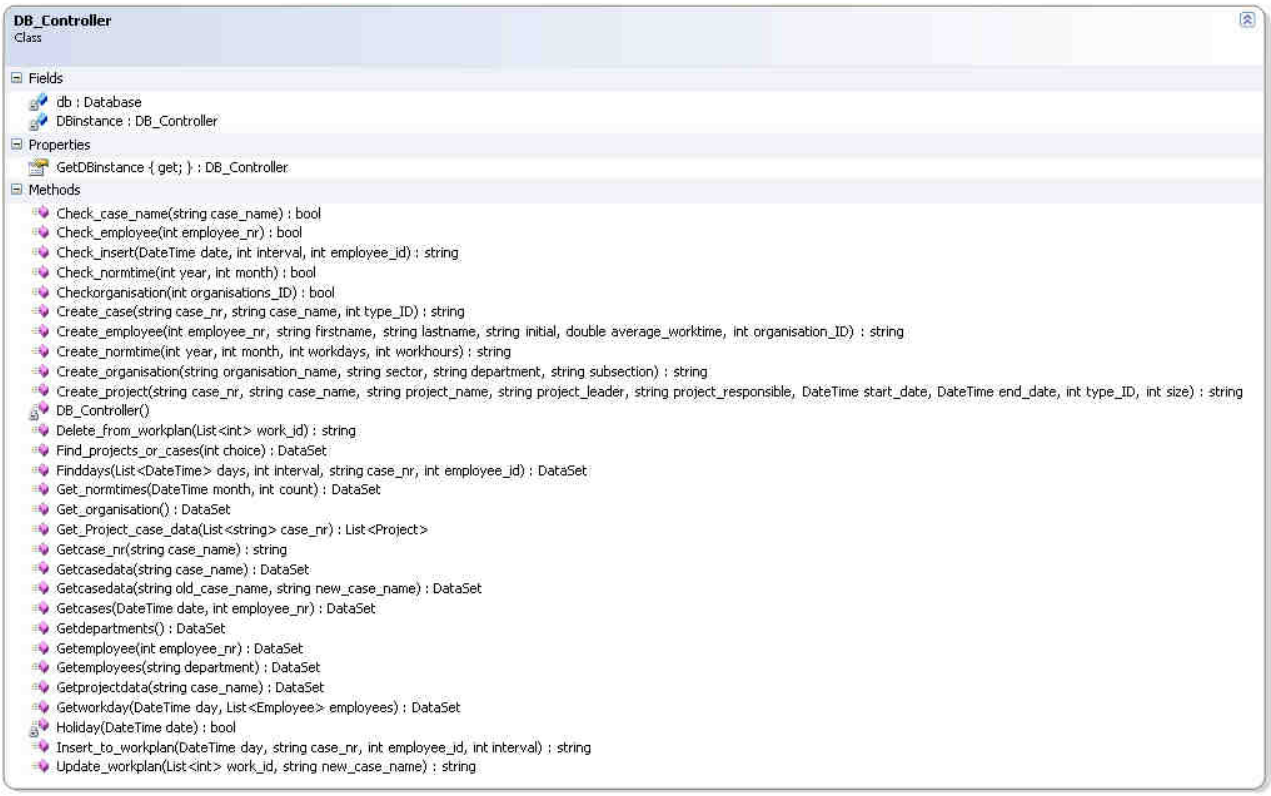


Figur 35 Organisation klassen

Denne klasse er ikke så vigtig på nuværende tidspunkt. Den bruges mest til at oprette en bestemt organisations del og til at hente data for organisationen såsom alle afdelinger.

6.1.8 DB_Controller klassen

Denne klasse styrer alt med database og er implementeret med de 2 beskrevet patterns fra design kapitlet. Den indeholder alle de nødvendige metoder til at hente, opdatere, slette og checke eksistens af udvalgte data i databasen.



Figur 36 DB_Controller klassen

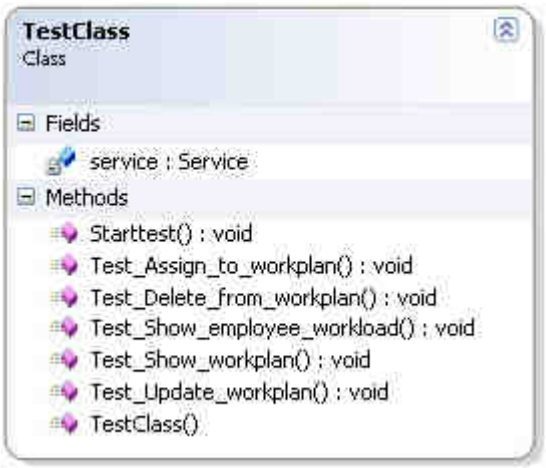
Vigtige metoder:

Navn	**Retur type**	**Parameter**	**Beskrivelse**
Holiday	bool	DateTime	Denne metode bruges til at teste om en bestemt dato, som den får som input, er en dansk helligdag. Den bruges af Insert_to_Workplan metoden, da det er den eneste metode, som indsætter noget i arbejdsplanen.. Den returnerer true/false alt efter om det er en dansk helligdag.
Create_project	string	string, string, string, string, string, DateTime, DateTime, int, int	Denne metode opretter et projekt i databasen ved at indsætte input parameterne i tabellerne sager og projekter. Da der skal udføres mere end en indsættelse, og de begge skal udføres samtidig eller fejle begge 2, laves indsættelsen, derfor vha. en transaktion. Her laves der en rollback, hvis den ene fejler, så data i databasen ikke vil

			være mangelfuld efter en indsættelse. Efter udførelsen returneres en meddelelse om det lykkes at udføre oprettelsen eller ej.
Insert_to_workplan	string	DateTime, string, int, int	Denne metode indsætter inputtet, som består af følgende data (medarbejdernummer, et sagsnr, dato og interval), i tabellen arbejdslisten, for at registre hvad medarbejderen skal arbejde med for en bestemt dato og interval.
Getworkday	DataSet	DateTime, List<Employee>	Metoden henter data fra arbejds_liste tabellen for hver medarbejder i listen for formiddag og eftermiddag for den pågældende dato, som er giver som input. Samtidig indsætter den også oplysninger om dagen er en weekend og om medarbejder er fri på dagen, hvis denne ikke er tildelt noget. Det DataSet som er lavet returneres med de fundet og tildelte værdier.

6.1.9 TestClass klassen

Denne klasse bruges udelukkende til test af de vigtigste webservice metoder i service klassen. Metoderne i denne klasse tester hver især en af metoderne fra Service klassen med forskellige inputs, såsom fejlinputs og grænse værdier, samt normale inputs[se afsnit 7.2].



Figur 37 TestClass klassen

Vigtige metoder:

Navn	Retur type	Parameter	Beskrivelse
Test_Assign_to_workplan	void	ingen	Denne metode bruges til at teste Assign_to_workplan metoden i Service klassen med forskellige input til denne. Inputtene består af normale værdier og fejl værdier, samt grænseværdier[se afsnit 7.2].
Test_Show_employee_workload	void	ingen	Denne metode bruges til at teste Show_employee_workload metoden i Service klassen med forskellige input til denne. Inputtene består af normale værdier og fejl værdier, samt grænseværdier[se afsnit 7.2].

6.2 Implementering af database

I dette afsnit vil jeg beskrive, hvordan jeg har implementeret databasen med hensyn til valg af feltnavne og typer for de enkelte tabeller, samt relationer mellem tabellerne, som kan ses i databasen diagrammet [se afsnit 6.2.2]. Databasen er blevet oprettet, som jeg beskrev i ER diagrammet under design kapitlet.

6.2.1 Tabellerne

Data typerne i de enkelte felter er valgt, så de passer med størrelse af data, som skal opbevares. F.eks. understøtter Int en størrelse på 2^16 bytes. Dvs. der kan være plads til et stort budget på et projekt. Det skal dog siges, at nogle data typer, hvor der står et Int kunne ændres til en mindre Int type, som f.eks. for at spare noget plads. Derudover er databasen opbygget, så de fleste felter ikke tillader nul værdier, eftersom det vil være plads spild.

Tabel: Medarbejder

Feltnavn	Data Type	Nøgler	Noter
Medarbejder_ID	Int, not null	Primær	Et nummer for medarbejderen, som er unikt for denne.
Fornavn	Varchar(50), not null		
Efternavn	Varchar(50), not null		
Initialer	Char(10), not null		Medarbejderens initialer eks. ”ELS”
Gnst_Arbejdstid	Decimal(2,0), not null		Gennemsnitsarbejdstid eks. En medarbejder arbejder 4 dage om ugen, dvs. denne har en Gnst_arbejdstid på 0.8.

Tabel: Projekter

Feltnavn	Data Type	Nøgler	Noter
Sags_Nr	Varchar(50), not null	Primær	Samme Sags_Nr som i tabellen Sager, da projekter er en del af en sag.
Projekt_Betegnelse	Varchar(50), not null		Navn på projektet
Projekt_Leder	Varchar(50), not null		
Ansvarshavende	Varchar(50), not null		Den person, som har det overordnede ansvar for projektet.
Start_Dato	Datetime, not null		Startdato for projektet
Slut_Dato	Datetime, not null		Slutdato for projektet
Størrelse	Int, not null		Bruges til at opbevare budget-størrelsen på projektet

Tabel: Normtider

Feltnavn	Data Type	Nøgler	Noter
Norm_ID	Int, auto genereret, not null	Primær	Et unikt id.

År	Int, not null		
Måned	Int, not null		
Antal_arbejdsdage	Int, not null		
Antal_arbejdstimer	Int, not null		

Tabel: Organisation

Feltnavn	**Data Type**	**Nøgler**	**Noter**
Organisations_ID	Int, auto genereret, not null	Primær	Et unikt id.
Organisation	Varchar(50), not null		Navn på en organisation
Sektor	Varchar(50), not null		Navn på en sektor
Afdeling	Varchar(50), not null		Navn på en afdeling.
Underafdeling	Varchar(50)		Navn på en underafdeling

Tabel: Arbejds_Liste

Feltnavn	**Data type**	**Nøgler**	**Noter**
Arbejds_Liste_ID	Int, auto genereret, not null	Primær	Et unikt id
Dato	Datetime, not null		
Sags_Nr	Varchar(50), not null	Fremmede	Sags_Nr for den sag en medarbejder er tildelt på denne dag og tid på dagen.
Medarbejder_ID	Int, not null	Fremmede	
Interval	Int, not null		Bruges til at bestemme tiden på dagen formiddag/eftermiddag

Tabel: Typer

Feltnavn	**Data Type**	**Nøgler**	**Noter**
Type_ID	Int, not null	Primær	Et unikt id
Type_Navn	Varchar(50), not null		
Produktivitets_Status	Bit, not null		Siger at en type er produktiv, hvis 1 ellers 0.

Tabel: Sager

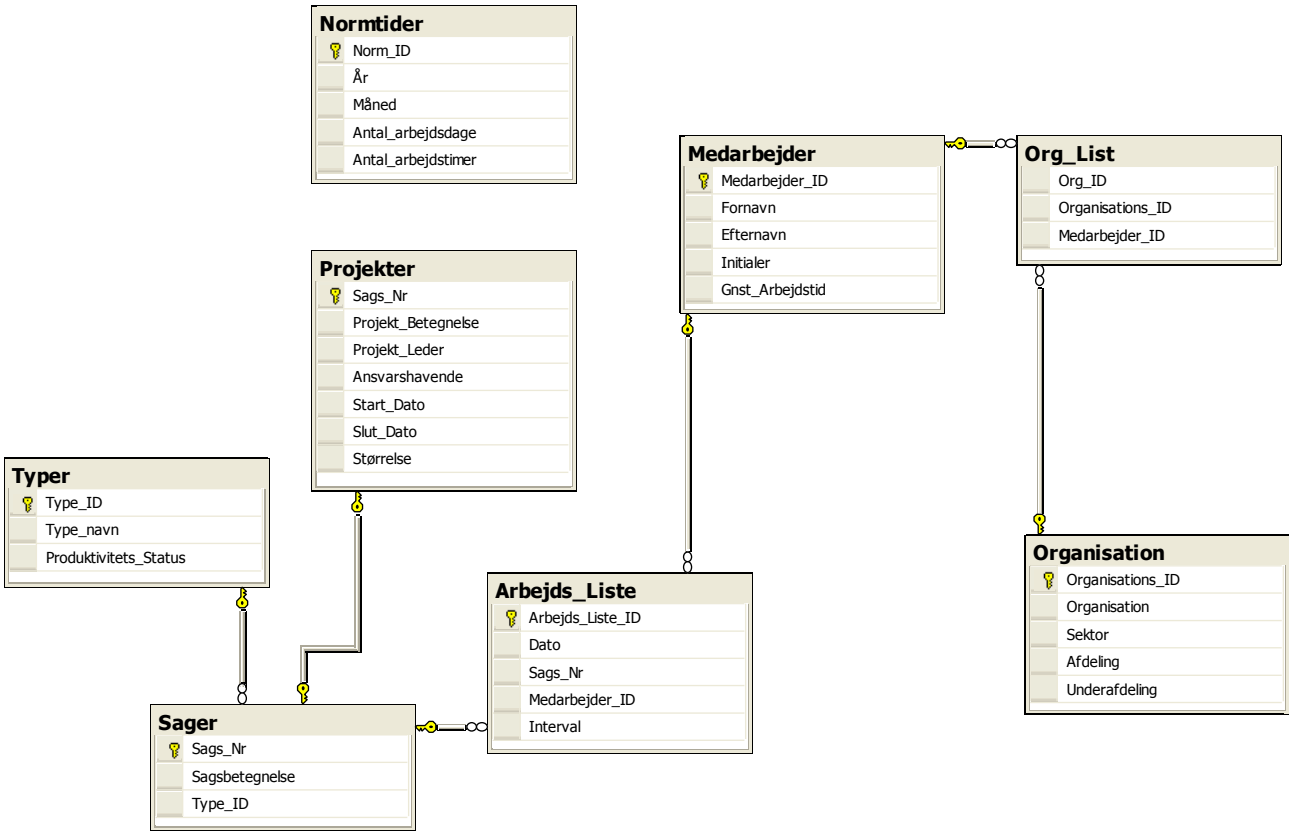
Feltnavn	Data Type	Nøgler	Noter
Sags_Nr	Varchar(50), not null	Primær	Samme Sags_Nr som i tabellen projekter, da en sag også kan være et projekt.
Sagsbetegnelse	Varchar(50) not null		Betegnelse for en sag.
Type_ID	Int, not null	Fremmede	

Tabel: Org_List

Feltnavn	Typer	Nøgler	Noter
Org_ID	Int, auto genereret, not null	Primær	Et unikt id
Organisations_ID	Int, not null	Fremmede	
Medarbejder_ID	Int, not null	Fremmede	

6.2.2 Database diagram

Database diagrammet er blevet lavet ud fra den oprettede database.



Figur 38 Database diagram for den oprettet database

6.3 Web implementation

For at kunne gøre brug af web servicens metoder i webapplikationen skal der addes tre xml filer. Filen WSDL er en xml fil, som definer hvordan interaktionen mellem webservicen og brugeren af denne skal foregå. Derud over er der en DISCO xml filen, som indeholder links til andre ressourcer som beskriver web service f.eks. link til WSDL filen. Den sidste fil er en DISCOMAP xml fil, som indeholder URL mapping til de 2 andre filer.
Disse filer laver Visual Studio selv, når man adder en web reference til web servicen og placerer filer i folderen App_WebReferernces. Disse dokumenter bruger webapplikation til at finde web servicen og hvordan interaktionen mellem de benyttede metoder er med hensyn til input og output parameter.
For at give et indtryk af, hvordan jeg har implementeret web service metoderne i webapplikationen, giver jeg her et eks. på brug af dem ved use casen ”Vis arbejdsbelastning for medarbejder”.
Man opretter en objekt reference for Service klassen, så man kan gøre brug af metoderne.

```
public partial class Employee_workload : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!IsPostBack)
        {
            DataSet ds;
            localhost.Service mess = new localhost.Service();
            ds = mess.Find_employees("GIS & IT");
            DataTableReader reader = ds.CreateDataReader();
            ListItem list;
            while (reader.Read())
            {
                list = new ListItem(reader[1].ToString(), reader[0].ToString());
                DropDownList1.Items.Add(list);
            }
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
            DateTime start_month = Calendar1.SelectedDate;
            localhost.Service ws = new localhost.Service();
            DataSet ds =
ws.Show_employee_workload(int.Parse(DropDownList1.SelectedValue), start_month);
            GridView1.DataSource = ds;
            GridView1.DataBind();
    }
}
```

Figur 39 Kode eks for brug af web service metoderne.

I dette eks. hentes først alle medarbejder initialer og medarbejder numre for medarbejderne i afdelingen ”GIS & IT”, som lægges i en dropdown liste, som brugeren kan bruge til at vælge en medarbejder ud fra. Når brugeren har foretaget sine valg og submittet det, kaldes metoden Show_employee_workload fra Service klassen med de valgte parametre. Derefter returneres et DataSet, som tilknyttes et Gridview control, som viser det hentede arbejdsbelastningsskema.

6.4 Resume

I implementationsdelen fik jeg beskrevet nogle af de vigtigste metoder i de enkelte klasser. Desuden fik jeg beskrevet, hvordan web applikationen gør brug af web servicens metoder. Derudover fik jeg beskrevet, hvordan jeg fik implementeret databasen med beskrivelse af de enkelte felter og deres typer, samt hvordan relationerne er imellem.

Kapitel 7

Test

7 Test

7.1 Indledning

For at teste udviklingen af programmet undervejs har jeg valgt at benytte manuelle tests, hvor jeg efter implementering af de enkelte use case laver en test af, om det virker, som det skal med hensyn til beskrivelse af, hvordan brugeren skal bruge systemet, og om han får det korrekte data vist.
Desuden vil jeg gøre brug af NUnit tests på nogle af de vigtigste web service metoder. Dette gøres for at verificere, om metoderne leverer det resultat, som jeg forventer ved forskellige inputs, og om der er nogle inputs, jeg ikke tager hensyn til. Herved kan jeg finde ud af, om der er nogle huller, som der skal findes løsninger på.

7.2 NUnit test

Da jeg ikke gør brug af Visual Studio Team Foundation Server, som har en del test værktøjer, har jeg inkluderet ”NUnit.Framework” i min web service del for at kunne gøre brug af de muligheder, dette giver mig til at lave mine egne Nunit tests for web service delen af projektet.
I og med at jeg kun har 10 uger til dette projekt, har jeg valgt at lave NUnit test for 5 af mine vigtige web service metoder.
Nunit testene giver mulighed for hele tiden at validere de metoder, som der er lavet en unit test over, så man f.eks. ved kodeændringer kan køre testene, og se om man stadigvæk får det forventede resultat ved forskellige inputs. De inputs man tester for i metoderne kan være mange, bl.a. er det normalt at teste for (normalværdier, grænseværdier, overgrænseværdier, null værdier, og forkerte indputs), derfor har jeg begrænset mig til følgende inputs.

- Normale indputs
- Forkerte indputs.
- Cheks på input værdier, som ikke findes i systemet.
- Overgrænseværdier bl.a. med visning af en arbejdsplan over et år.

Det første jeg gør for at oprette en Nunit test af en metode er at lave en beskrivelse af testen med testbetingelser for input, og hvad resultatet af testen forventes, at være med det pågældende input

```
/// <summery>
/// This method test the method Assign_to_workplan()
/// returns a message
/// Precoditions:
/// The relavant data is in the database or not in the database.
/// The test consist of the following test conditions.
/// 1. A normal test with an existed employee, case/projekt, normal dates interval.
/// 2. A test with a not existed employee.
/// 3. A test with a not existed case.
/// 4. A test with an allocation that existed a projekt period.
/// 5. A test where data already exist in the choosen period.
/// 6  A test with a allocation over a period of 3 months
/// 7. A test with a enddate before a startdate
/// </summery>
/// <remarks>
/// - Desired Result: A message "Sag blev tildelt til medarbejder: xxxx"
/// - Coordinator: When
/// - Conditions:  1
///
/// - Desired Result: A errormessage: "Medarbejder xxxx findes ikke"
/// - Coordinator: When
/// - Conditions:  2
///
/// - Desired Result: A errormessage: "Sags_Betegnelse findes ikke"
/// - Coordinator: When
/// - Conditions:  3
///
/// - Desired Result: A errormessage: "Interval er ikke i projekt perioden"
/// - Coordinator: When
/// - Conditions:  4
///
/// - Desired Result: A errormessage: "Ikke muligt at indsætte"
/// - Coordinator: When
/// - Conditions:  5
///
/// - Desired Result: A message: "Sag blev tildelt til medarbejder: xxxx"
/// - Coordinator: When
/// - Conditions:  6
///
/// - Desired Result: A errormessage: "Slutdato angivet før startdato"
/// - Coordinator: When
/// - Conditions:  7
/// </remarks>
```

Figur 40 Eksempel på test beskrivelsen af Assign_to_workplan()

Bagefter laves selve testmetoden, som tester de forskellige konditions og gør brug af de metoder ”NUnit Framework” har, såsom Assert.AreEqual(”Forventet resultat”, ”output fra metoden”), som kaster en exception, hvis de 2 strenge ikke er ens. Herved fejler betingelsen, og en fejlbesked skrives til konsol vinduet.

```
// Test condition 1
    message = service.Assign_to_workplan(new DateTime(2007,9,1),new DateTime(2007,9,20),2,90000,"Ferie");
    try
    {
       Assert.AreEqual("Sag blev tildelt til medarbejder: 90000",message);
       Debug.WriteLine("1. Normal test: Succes");
    }
    catch (AssertionException)
    {
       Debug.WriteLine("1. Normal test: Failed");
    }
```

Figur 41 Uddrag af koden fra Test_Assign_to_workplan() med test af kondition1

Således testes hver kondition i de forskellige testmetoder. Før selve testen kan udføres, kaldes en SetupTest() metode, som sørger for at oprette det nødvendige test data, hvis det ikke allerede er i databasen.

```
public void SetupTest()
{
    string message;
    // Creates an new test employee.
    message = service.Create_employee(90000, "Henrik", "Andersen", "HEA", 1, 1);
    message = service.Create_case("2/222", "Ferie", 5);
    message = service.Create_case("2/3333", "NUnitTest", 5);
    message = service.Create_project("2/testcase", "Testprojekt", "Bigtestprojekt", "ELS", "JSC", new
    DateTime(2008, 1, 10), new DateTime(2008, 4, 10), 1, 300000);
    message = service.Delete_from_workplan(new DateTime(2009, 7, 1), new DateTime(2009, 9, 1), 2, 90000,
    "NUnitTest");
}
```

Figur 42 SetupTest() metoden som kaldes før testen køres

Når testen kører, kaldes alle testmetoderne, og de tester hver især deres repræsentive metode med forskellige konditionens og skriver til konsolen med resultatet af testen.

```
Output
Show output from: Debug
Test of Assign_to_workload()
1. Normal test: Succes
2. Test no employee: Succes
3. Test no case: Succes
4. Test project periode: Succes
5. Test allocation on a already booked date: Succes
6. Test allocation over a long period: Succes
7. Enddate before startdate: Succes
Test of Update_workplan()
1. Normal test: Succes
2. No cases: Succes
3. No employee: Succes
4. Update over a long period: Succes
5. Update old_casename that don't exist in period: Succes
6. Update with enddate before startdate: Succes
Test of Delete_from_workplan()
1. Normal delete test: Succes
2. No employee: Succes
3. No case Succes
4. Delete in periode with no data: Succes
5. Delete over a long period: Succes
6. Enddate before startdate: Succes
Test showworkplan()
1. Normal date test: Succes
2. Dates over an year: Succes
3. Endmonth before startmonth: Succes
4. Department doesn't exsist or have no employees: Succes
Test af show_employee_workload()
1. Normal test: Succes
2. Test with no cases assigned: Succes
3. Test with no normtimes: Succes
4. Test with nonexsisten employee: Succes
```

Figur 43 Dump af testkørsel af Nunit testes, hvor alle test udføres med succes.

Da jeg lavede de her test, fandt jeg nogle fejl, som jeg fik rettet, bl.a. var der problemer med tildeling af en sag/projekt til en medarbejder over en længere periode, da en tæller i en løkke talte en for meget op. Desuden kan de her test også bruges til løbende at teste metodernes virkning efter en udbygning af koden. Derfor ville jeg have lavet Nunit test på flere af metoderne, hvis der havde været tid til det.

7.3 Manuel test

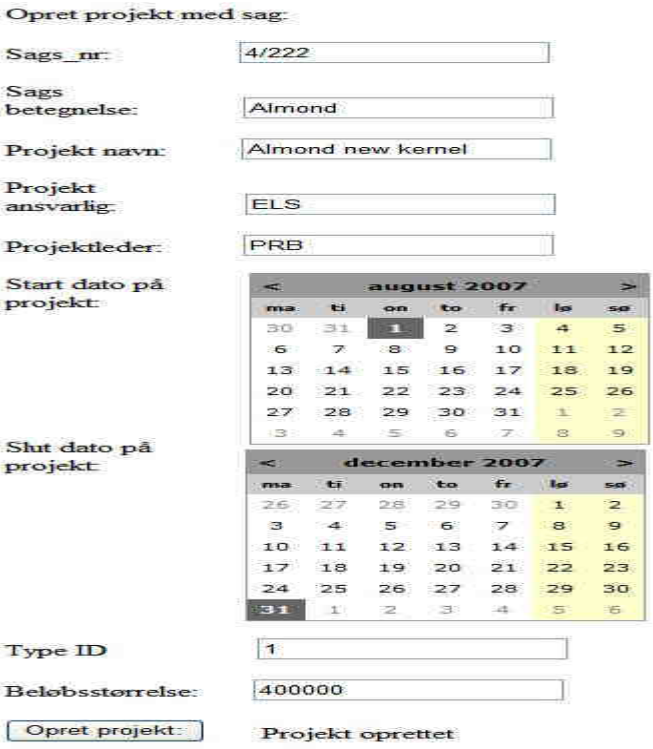
Manuelle tests er blevet udført i slutningen af hver iterations for at verificere om de implementerede use case virkede, som de skulle med hensyn til brugerens interaktion med systemet, eller om der var fejl, som eventuel skulle rettes inden starten af næste iteration.
Jeg vil i dette afsnit vise et eksempel på en manuel test i webapplikationen og i service delen, hvor følgende laves.

- Opretter en sag og et projekt.
- Opretter en medarbejder
- Tildeler sagen og projektet til en medarbejder for en periode.
- Ser arbejdsplanen for medarbejderens afdeling.
- Ser arbejdsbelastningsskemaet for medarbejderen
- Opretter en ny normtid laves i web service delen, da den ikke er implementeret i webdelen.

Først prøves at oprette en sag.

Figur 44 Sag oprettes

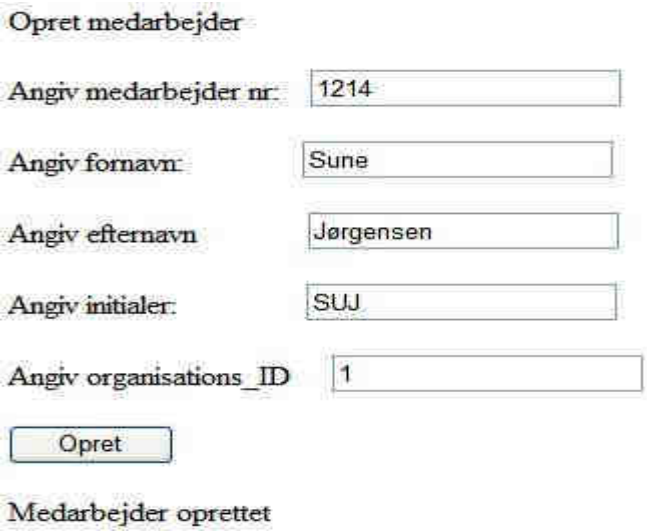
Sagen blev oprettet. Hvis man vælger et allerede eksisterende sagsnr, sagsbetegnelse eller et type ID, som ikke findes, vises en fejlmeddelelse.
Herefter prøves at oprette et projekt.



Figur 45 Projekt oprettes

Ved projekt oprettelsen gives også en fejlmeddelelse, hvis man angiver et allerede eksisterede sagsnr, sagsbetegnelse eller type id, samt angiver en slutdato før en startdato.

Herefter prøves at oprette en medarbejder.



Figur 46 Medarbejder oprettes

Ved medarbejder oprettelsen gives en fejlmeddelelse, hvis man angiver et allerede eksisterede medarbejder Nr eller et organisations id, som ikke findes.

Nu har systemet 2 sager og en medarbejder, så derfor tildeler vi nu disse til medarbejderens arbejdsplan for nogle bestemte perioder.

Figur 47 Sag tildeles til medarbejders arbejdsplan

Ved tildelingen af sagen til medarbejderen gives en fejlmeddelelse, hvis medarbejder allerede er tildelt noget i den valgte periode, eller hvis slutdato er angivet før en startdato. Desuden gives ved tildelingen af et projekt en fejlmeddelelse, hvis angivet periode ligger ud over projektperioden

Figur 48 Projekt tildeling udenfor projektperioden

For at se om medarbejderen er blevet tildelt sagen korrekt i perioden, vælges at se på arbejdsplanen for afdelingen ”GIS & IT”, som medarbejderen er tilknyttet og månederne februar og marts, som medarbejderen blev tildelt sagen i.

Angiv afdeling:
GIS & IT
Angiv start start måned: (åååå/mm)
2008/Februar
Angiv slut måned: (åååå/mm)
2008/Marts
Hent

Tilbage Frem

Dato	TOB	TIB	MOB	KAM	STE	TAD	SIH	PEA	HEA	SUJ
For: 1/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 1/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 2/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 2/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 3/3/2008					Ferie				Ferie	Afspadsering
Eft: 3/3/2008					Ferie				Ferie	Afspadsering
For: 4/3/2008					Ferie				Ferie	Afspadsering
Eft: 4/3/2008					Ferie				Ferie	Afspadsering
For: 5/3/2008					Ferie				Ferie	Afspadsering
Eft: 5/3/2008					Ferie				Ferie	Afspadsering
For: 6/3/2008					Ferie				Ferie	Afspadsering
Eft: 6/3/2008					Ferie				Ferie	Afspadsering
For: 7/3/2008					Ferie				Ferie	Afspadsering
Eft: 7/3/2008					Ferie				Ferie	Afspadsering
For: 8/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 8/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 9/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 9/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 10/3/2008					Ferie				Ferie	Afspadsering
Eft: 10/3/2008					Ferie				Ferie	Afspadsering
For: 11/3/2008					Ferie				Ferie	Afspadsering
Eft: 11/3/2008					Ferie				Ferie	Afspadsering
For: 12/3/2008					Ferie				Ferie	Afspadsering
Eft: 12/3/2008					Ferie				Ferie	Afspadsering
For: 13/3/2008					Ferie				Ferie	Afspadsering
Eft: 13/3/2008					Ferie				Ferie	Afspadsering
For: 14/3/2008					Ferie				Ferie	Afspadsering
Eft: 14/3/2008					Ferie				Ferie	Afspadsering
For: 15/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 15/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 16/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft: 16/3/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For: 17/3/2008					Ferie				Ferie	

Figur 49 Arbejdsplan for marts

Ved at klikke frem til marts måned kan vi se, at medarbejderen SUJ er blevet tildelt sag afspadsering korrekt ved kun at blive tildelt de normale arbejdsdage, som fandtes i perioden.

Eft. 17/2/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For. 18/2/2008					Ferie				Ferie	Afspadsering
Eft. 18/2/2008					Ferie				Ferie	Afspadsering
For. 19/2/2008					Ferie				Ferie	Afspadsering
Eft. 19/2/2008					Ferie				Ferie	Afspadsering
For. 20/2/2008					Ferie				Ferie	Afspadsering
Eft. 20/2/2008					Ferie				Ferie	Afspadsering
For. 21/2/2008					Ferie				Ferie	Afspadsering
Eft. 21/2/2008					Ferie				Ferie	Afspadsering
For. 22/2/2008					Ferie				Ferie	Afspadsering
Eft. 22/2/2008					Ferie				Ferie	Afspadsering
For. 23/2/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft. 23/2/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For. 24/2/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
Eft. 24/2/2008	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend	Weekend
For. 25/2/2008					Ferie				Ferie	Afspadsering
Eft. 25/2/2008					Ferie				Ferie	Afspadsering
For. 26/2/2008					Ferie				Ferie	Afspadsering
Eft. 26/2/2008					Ferie				Ferie	Afspadsering
For. 27/2/2008					Ferie				Ferie	Afspadsering
Eft. 27/2/2008					Ferie				Ferie	Afspadsering
For. 28/2/2008					Ferie				Ferie	Afspadsering
Eft. 28/2/2008					Ferie				Ferie	Afspadsering
For. 29/2/2008					Ferie				Ferie	Afspadsering
Eft. 29/2/2008					Ferie				Ferie	Afspadsering

Figur 50 Arbejdsplan for februar

Her kan vi se, at medarbejderen SUJ er blevet tildelt Afspadsering for de korrekte dage i februar, samt at der er taget hensyn til skudåret i 2008 ved, at det er muligt at arbejde den 29. februar.

For at se hvor hårdt medarbejderen er belastet et år frem i tiden, vælges at se arbejdsbelastningsskemaet for denne.

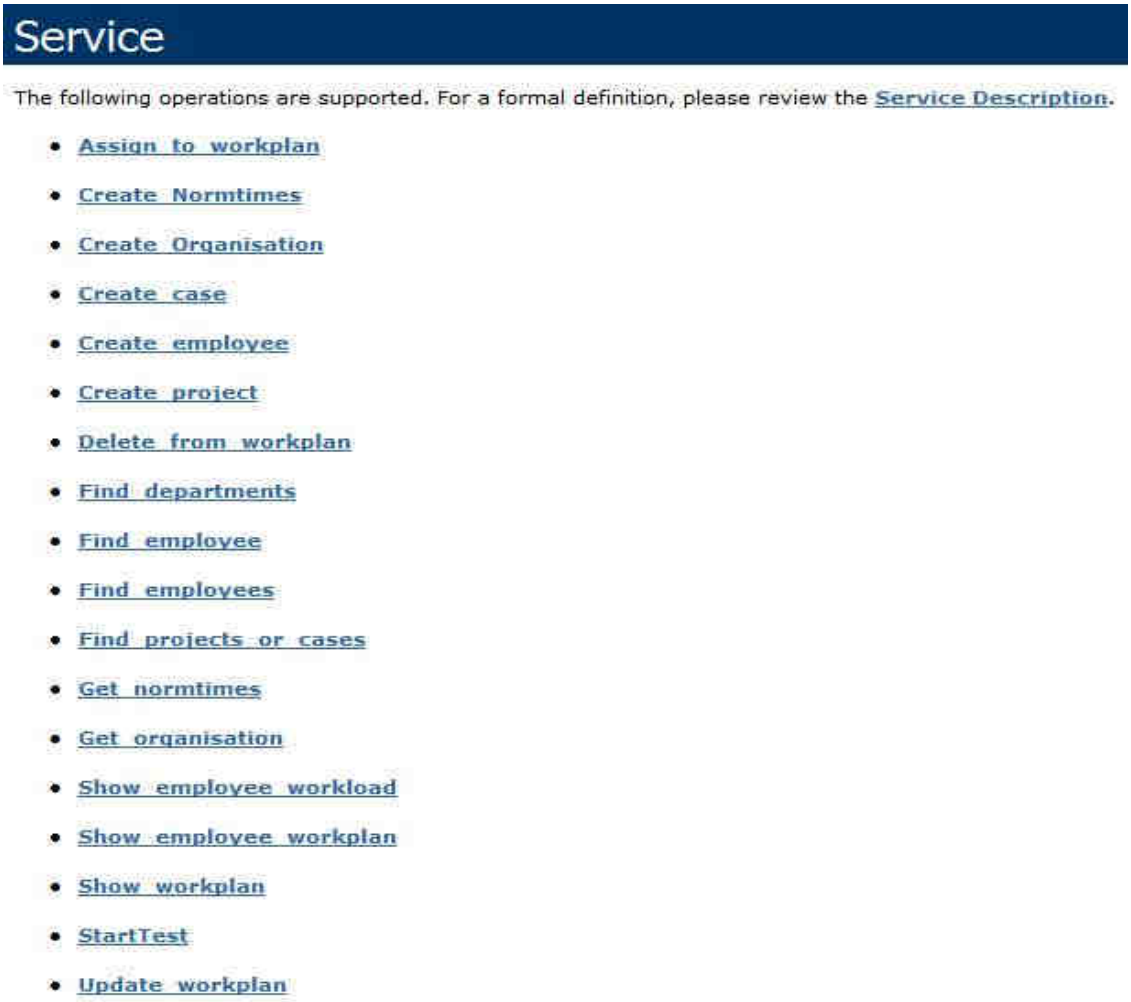
Figur 51 Vælger at se arbejdsbelastningen for medarbejderen SUJ fra oktober og et år frem

Type	Sags Nr	Sags_navn	PI/PGL	Oct	Nov	Dec	Jan	Feb	Mar	Apr	Maj	Jun	Jul	Aug	Sep	Ialt
IGA større projekter >= 200000 kr:	3151	Asset Mangement	ELS/ELS	170	15	0	0	0	0	0	0	0	0	0	0	185
	4/222	Almond	PRB/ELS	0	18	18	0	0	0	0	0	0	0	0	0	36
			SUM	170	33	18	0	0	0	0	0	0	0	0	0	221
IGA større projekter < 200000 kr:																
			SUM	0	0	0	0	0	0	0	0	0	0	0	0	0
Interne opgaver (adm/salg/tilbud):																
			SUM	0	0	0	0	0	0	0	0	0	0	0	0	0
Kommende kendte projekter:																
			SUM	0	0	0	0	0	0	0	0	0	0	0	0	0
Forventede projekter fra tilbud/salg:																
			SUM	0	0	0	0	0	0	0	0	0	0	0	0	0
Diverse(Ferie/barsel/orlov/afspad):	3/7722	Afspadsering		0	0	0	0	74	74	0	0	0	0	0	0	148
			SUM	0	0	0	0	74	74	0	0	0	0	0	0	148
Total timebelasning			Total	170	33	18	0	74	74	0	0	0	0	0	0	369
Månedens time kapacitet:				170	163	141	163	155	133	170	148	155	170	155	163	1886
Månedens belastning:				100%	20%	12%	0%	47%	55%	0%	0%	0%	0%	0%	0%	
Månedens forventede uddebitering:				100%	20%	12%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
Månedens sikre uddebitering:				100%	20%	12%	0%	0%	0%	0%	0%	0%	0%	0%	0%	

Figur 52 Arbejdsbelastningsskemaet for medarbejderen SUJ

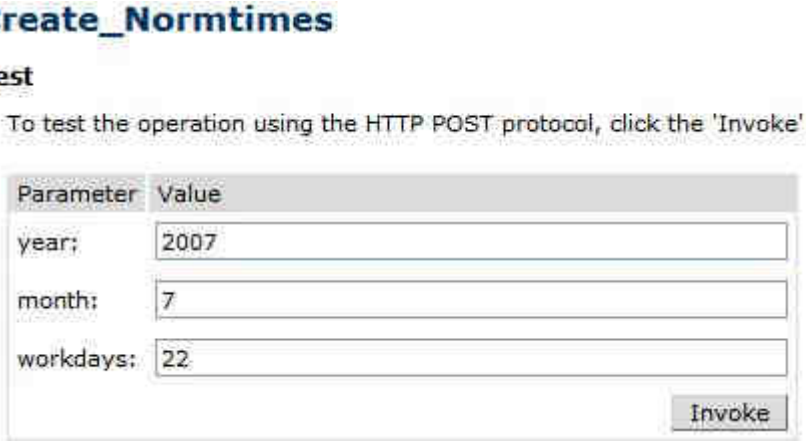
Her kan vi se den pågældende medarbejders arbejdsbelastning med uddebiteringsgrader og timebelastning for hver måned i det valgte år. De 10 arbejdsdage, vi tildelte både i februar og marts til afspadsering, angiver at brugeren bruger 74 arbejdstimer hver måned på at afspadsere, og at han har en timebelastning på henholdsvis 47 % og 55 % for hver af de måneder i forhold til månedernes arbejdstimer.

Desuden er begge hans uddebiteringsgrader for de 2 måneder 0 %, eftersom afspadsering hører under typen Diverse, og der derfor ikke kan laves en fakturer til en kunde. Hvis man til gengæld kigger på oktober, er medarbejderen beskæftiget med noget alle arbejdsdagene i måneden og har derfor en belastningsgrad på 100 %, samt 2 uddebiteringsgrader på 100 %, eftersom Asset Management er et projekt og har type 1. Det vil sige, at der kan laves en fakturer ud til kunde på de 170 arbejdstimer i måneden. Ved at se skemaet for andre perioder, kan systemet returnere en fejlmeddelelse, hvis medarbejderen ikke er tildelt nogle sager/projekter i de 12 måneder:
Samtidig laves der heller ikke nogen beregninger for de måneder, som ikke har nogen normtider i systemet. Her skrives der et nul i månedens time kapacitet for hver af de måneder.



Figur 53 De metoder web servicen tilbyder, og som webapplikationen bruger nogle af

Da ikke alle metoder bruges i test webapplikation, kan Web servicen give brugeren mulighed for at benytte dem.



Figur 54 Vælger at oprette en ny normtid for et år og måned

Figur 55 web servicens svarer, at oprettelse lykkes

Hvis oprettelsen ikke kunne lade sig gøre, returneres en fejlmeddelelse om dette. Det kan enten være, fordi der allerede findes normtid for den angivet måned, eller fordi indsættelsen ikke kunne lade sig gøre.

7.4 Resume

Jeg fik i dette afsnit forklaret lidt om valg af test under mit projekt, og hvordan jeg har brugt dem. Samtidig fik jeg vist eksempler på min brug af Nunit og manuelle tests. Ved kørslen af de manuelle test i slutningen af hver iteration, fandt jeg fejl, som jeg enten fik rettet samtidig eller inden start af næste iteration. I starten af min implementation drejede det sig især om databasen's opbygning. Efterhånden som der blev gennemført flere iterationer, faldt fejlprocenten. Nunit testenes testbeskrivelser gav mig indblik i mulige fejltyper ved inputs til metoderne, som jeg skulle tænke på under mit design og implementation.
Samtidig kunne jeg køre dem løbende, og se om ændringer i koden gjorde, at testene fejlede, så jeg kunne finde fejlene hurtigt og få dem rettet.

Kapitel 8

Konklusion

8. Konklusion

8.1 Indledning

Formålet med dette projekt var at lave et webbaseret ressource allokeringssystem til projektplanlægning for afdelingen ”GIS & IT”, men som også vil kunne bruges af andre afdelinger hos Atkins Danmark. Projektet skulle tage udgangspunkt i data fra de 2 nuværende benyttede systemer Blast 32 og Excel ark. Systemet vil give muligheden for at allokere sager/projekter til en medarbejders arbejdsplan.
Systemet kan så bruge denne allokering til at få et bedre overblik over den enkelte medarbejders arbejdsbelastning, samt arbejdsplanen for medarbejdere i afdelingen. Herved ville man bedre kunne planlægge kommende projekter med hensyn til, hvilke medarbejderressourcer man har til rådighed.

8.2 Resume

Dette projekt blev udviklet efter UP i en iterativ proces, hvor systemets use case gradvis blev implementeret, så system gradvist fik mere funktionalitet.
Først blev selve kravene(use cases) til det nye system fundet via interviews med aktørerne, samt et kig på de 2 nuværende systemer. Derudover blev der lavet en risikoplan, som afgjorde hvilke ting, som der skulle tages fat på i de første iterationer.

En analyse af de enkelte use cases blev lavet med use case beskrivelser og system sekvens diagrammer, som beskrev brugerens brug af systemet. Desuden blev en domain model lavet, som gav et overblik over konceptet for system.

I hver iteration blev et design lavet for hver af de valgte use cases med sekvens diagrammer og et klasse diagram, som gradvist blev udvidet. Desuden blev et design lavet over den mulige opbygning af databasen.

I implementation blev databasen og de valgte use cases i hver iteration ud fra designet implementeret. Alle de valgte use cases blev implementeret i web service delen, mens en række af use casene også blev implementeret i den webapplikation, som skulle gøre brug af de implementerede web service metoder. Når den endelige webapplikation skal laves engang efter dette projekt vil alle use casene indgå.

Efter implementation af use casene i hver iteration blev der lavet en manuel test, som testede om use casene leverede det, som brugeren forventede af den. Desuden blev der lavet en række Nunit test på nogle af de vigtigste web service metoder [Se afsnit 7.2 Nunit test].

Alle use casene på nær UC 13, UC 12, og UC 3[se afsnit 3.3.3 Use case diagram] blev i dette projekt implementeret. UC 3 udgik, eftersom man allerede kan se frie ressourcer i arbejdsplan for medarbejdere. UC 12 og UC 13 kan man allerede se i systemet blast 32, derfor vil de blot være en ekstra feature til dette planlægningssystem.
Det færdige system for dette projekt består af en SQL server 2005 database, som indeholder alt det nødvendige data til planlægningssystemet, samt en web service der bruger denne database og som indeholder alle de nødvendige rutiner/algoritmer, som der er brug for. Samtidig er der blevet implementeret en simpel webapplikation, som bruger nogle af web servicen metoder til arbejdsplanlægningen.

8.3 Evaluering

Jeg fik lavet en prototype på et webbaseret ressourceallokeringssystem til projektplanlægning, som kan hjælpe de enkelte afdelinger hos Atkins med at give et overblik over hvad de enkelte medarbejdere skal arbejde med frem i tiden, så medarbejderressourcer til projekter bedre kan planlægges. Desuden giver systemet mulighed for at tildele/opdatere/slette sager/projekter på en medarbejders arbejdsplan. I forhold til systemet med excel ark, er det nu muligt at lagre oplysninger om sager/projekter, medarbejdere og normtider i en database, i stedet for manuelt at skulle taste det ind hver gang man skal ændre i arkene. Desuden kan man i arbejdsplanen for det nye system tildeles mere end en sag eller et projekt pr dag.
Samtidig skal man heller ikke ind og ændre i arbejdsbelastningsskemaet hele tiden, da disse nu bliver lavet dynamisk, så det tilpasses den enkelte medarbejders arbejdsplan. I forhold til systemet Blast 32 vil det også være muligt at lave de samme summeringer over medarbejder kapacitet, som bliver gjort her, da nødvendige data er til rådighed. Samtidig vil man også kunne lave udtræk for enkelte projekter og se de tilknyttede medarbejderes arbejdstimer på disse. Dette er ikke blevet lavet i dette projekt, eftersom man allerede kan se dette i Blast 32, men kan laves i en videre bygning af dette projekt. Da Blast 32 er et system, som ikke er særligt fleksibelt og nemt at finde rede i, ville det være en fed feature at få disse funktioner integreret i dette projekt senere hen.

I og med at jeg har lavet en Web service, som har de nødvendig rutiner/algoritmer til planlægningen, vil det blive lettere at lave flere webapplikationer i fremtiden. Da web servicen har alt logikken og database tilgangen, skal webapplikationen bare sørge for brugere interfacet.

De Nunit teste, som jeg fik lavet vil også i videre udvikling af programmet kunne bruges til løbende at evaluere, om metoderne leverede det forventede resultat ved forskellige input ved ændringer i koden. Desuden vil det være smart at lave flere af disse i videreudvikling af dette projekt.

UP gjorde min opgaver lettere at lave, da jeg hurtigt kunne finde kravene til system og i hver iteration have et produkt at forholde mig til. Desuden kunne nye krav lettere integreres i opgaven undervejs.

Der er stadig en række ting, som kunne forbedres i det nuværende projekt. Selve kodningen skal optimeres, så lav kobling understøttes. F.eks. er det ikke meningen at Employee_report klassen skal kende til Project klasse, dette kunne løses ved at Service klassen henter en liste med Project objekterne og sender dem videre til Employee_report klassen, derved undgås en association mellem de Project klassen og Employee_report klassen.
Desuden vil det være smart, hvis opdatering og tildeling af projekter/sager til en medarbejders arbejdsplan bliver lavet med en transaktion ligesom opret medarbejder og projekt, da man laver flere indsættelser her. De er dog ikke så kritiske her, da data i databasen ikke vil blive mangelfuld ligesom med opret medarbejder og projekt. Men det vil gøre systemet bedre, eftersom man ellers skulle ind og opdatere eller tildele igen for de dage, som ikke blev indsat, hvis der sker en fejl under indsættelsen.

8.4 Perspektiv og fremtidige udvidelser

Webapplikationen vil blive udbygget senere i iterationer efter dette projekt. Samtidig vil selve systemet også blive sat rigtigt i produktion, så fremtiden for dette projekt vil være, at afdelingerne i sektor Miljø & GIS benytter sig af systemet til at holde styr på, hvad medarbejderne arbejder med, og hvad deres belastninger, så kommende projekter bedre kan planlægges med hensyn til medarbejderressourcer. Derudover kan det tænkes, at flere afdelinger i de andre sektorer hos Atkins Danmark også vil benytte sig af systemet.

Med hensyn til fremtidige udvidelser og rettelser kan jeg forestillede mig følgende:

- **Database konsistens** - Alle indsættelser/opdateringer i databasen, som kræver flere indsættelser, skal laves med transaktioner, så der kan laves en ”Rollback”, hvis der sker en fejl..
- **Kørsel** – Koden kan effektiveres ved hentning af arbejdsplanen, så der ikke hentes data for hver dag, men for hver måned i stedet for. Desuden skal koden rettes til, så den understøtter lav kobling mellem klasserne.
- **Webapplikation –** Skal laves med login, så kun projektleder kan få adgang til at redigere en medarbejders arbejdsplan. Desuden skal webapplikationen laves med et bedre design.
- **Web servicen** – De sidste use cases skal implementeres for at give de ekstra features. Desuden kan der komme krav om nye funktioner, såsom sletning af sager/projekter og medarbejdere, samtidig kunne hentning af sager/projekter specificeres, så man kun henter nogle ud fra bestemte kriterier. Gennemsnits arbejdstiden for en medarbejder skal også laves, så der checks på om en medarbejder max må arbejde 4 dage om uge ved tildeling.
- **Data fra økonomi system** – I og med at Atkins får nyt økonomi system efter nytår, kunne det være godt at hente den nødvendige data for sager/projekter og medarbejdere herfra.

Litteraturliste

9. Litteraturliste

1. Applying UML and Patterns
   An introduction to Object-Oriented Analysis and
   Design and Iterative Development
   Third Edition
   Craig Larman
2. http://users.skivehs.dk/ob/database_04f/public_html/data02.html
3. http://www.hedeskov.dk/netpublikationer/udv-database/kap2/kapitel2.htm
4. http://dotnetjunkies.com/Article/29EF3A4F-A0C2-4BB2-A215-8F87F100A9F9.dcik
5. http://aspnet.4guysfromrolla.com/articles/030905-1.aspx

Bilag

Bilag A

Interview angående nyt planlægningssystem

I forbindelse med at finde behovene til projektplanlægningssystemet er det vigtigt at få ønsker/behov for den gruppe af aktører, som kunne tænkes at gøre brug af sådan et system i fremtiden. Dette har jeg valgt at gøre vha. interviews med aktørerne og ud fra generelle behov. Systemet kan herved tage hensyn til de vigtigste af disse behov og give et projektplanlægningssystem, som er bedre end de nuværende 2 systemer.
Til repræsentation af disse aktører vælges en afdelingsleder, en projektleder og en almindelig medarbejder, som skal udgøre interview grundlaget. Hvis der havde været mere tid til dette, havde interview grundlaget selvfølgelig været større.

Interview medarbejder	Herdis Gudbrandsdottir
Spørgsmål 1:	Hvad kunne dine behov være til et nyt web baseret planlægningssystem i stedet for det nuværende med excel regneark?
Svar:	• Det ville være godt, hvis man kunne have flere projekter/sager på en dag i arbejdsplanen. • Det vil samtidig være godt, hvis man kunne få et overblik over sin egen arbejdsplan uden at skulle se andres planer. • Dog vil det selvfølgeligt være rart, hvis man stadig kunne vælge at se andres tidsplaner, da man herved kan planlægge samarbejde med andre, hvis de arbejder med noget, der ligner. • Desuden skal man kunne vælge en periode, man ønsker at se arbejdsplanen for, så man herved bedre kan få et overblik over, hvad man skal lave i fremtiden.
Spørgsmål 2:	Hvad vil du selv have lyst til at gøre med din arbejdsplan i sådan et system?
Svar:	• Jeg vil bruge den til at give mig overblik over, hvad jeg skal arbejde med i en valgt periode. • Det vil ikke være nødvendigt selv at taste ind, hvad der

	arbejdes på i den periode. Det vil projektlederen gøre.

Interview Projektleder	Anne Jacobsen
Spørgsmål 1:	Hvilke Behov vil du have til et nyt web baseret planlægningssystem i stedet for det nuværende med arbejdsplan og arbejdsbelastning i excel regneark?
Svar:	• Der skal være mulighed for at se en arbejdsplan for alle medarbejdere i en afdeling med hvilken projekter/sager, de er tilknyttet på dagsbasis. Her skal der ikke gåes længere ned i detaljegrad end en halv dag, så man f.eks. kan have en halv feriedag og en halv arbejdsdag på et projekt eller en sag skrevet på for dagen. • Man skal i forhold til den nuværende plan også kunne føre tilbuds arbejde ind, så det bliver taget med i beregningerne af uddebiteringsgrader i medarbejderbelastningsskema. • Det skal kun være projektledere, som skal kunne redigere og tildele projekter/sager på en medarbejders arbejdsplan, for ellers ville det hurtigt kunne blive uoverskueligt. Den enkelte medarbejder skal så henvende sig til den relevante projektleder for at få rettet i sin arbejdsplan. • Planen skal også kunne give plads til frie dage, hvor man ikke er tildelt noget, da man jo ikke altid arbejder med noget, som kan skrives på. • Det kunne også være godt, hvis det var muligt at se hvem, som har fortaget ændringer sidst i en medarbejders arbejdsplan via noget logning.
Spørgsmål 2:	Hvilke ting kunne du tænke dig at få et overblik over i sådan et system.
Svar:	• Systemet skal kunne give mig et overblik over arbejdsplanen frem i tiden for medarbejderne i en afdeling, så man bedre kan

	planlægge med hvilke ressourcer, man har til rådighed til nye projekter frem i tiden.. • Det kunne ligesom i de nuværende ark være godt stadigvæk at kunne se medarbejderbelastning frem i tiden pr måned.

Interview afdelingsleder:	Eli Skoop
Spørgsmål 1:	Hvilke behov vil du have til et nyt webbaseret planlægningssystem i stedet for det nuværende med Blast 32?
Svar:	• Der skal være mulighed for at kunne gå ind under organisations niveau(sektor, afdeling) og få en prognose for arbejdskapacitet fordelt med mænd timer på de forskellige typer af projekter/sager, som de arbejder på. Den skal kunne vise måneder frem i tiden. • Prognose skal fortages på baggrund af data i arbejdsplanen.
Spørgsmål 2:	Hvilke ting kunne du tænke dig at få et overblik over i sådan et system.
Svar:	• Den nævnte arbejdskapacitet skal være inddelt efter projekt typer, og den skal gerne vises grafisk. • Desuden kunne det være godt, hvis man kunne få et overblik over den enkelte medarbejders belastning måneder frem i tiden fordelt på projekter/sager.

Bilag B

Evaluering over 1 iteration:

I den første iteration blev databasen for arbejdsplanlægningen oprettet først. Herefter blev use casen ”Vis arbejdsplan for medarbejdere” lavet med design, implementation og test. Under forløbet med denne use case er det fundet nødvendigt at ændre på designet for databasen inden starten af 2 iteration. Følgende ændringer er blevet vedtaget i forhold til det oprindelige design af databasen.

a. Der bliver oprettet en tabel Org_Liste, som skal holde medarbejder_ID og Organisation_ID, fordi en medarbejder herved kan være tilknyttet flere afdelinger eller flere sektorer i stedet for den gamle måde med, at organisation_ID’et lå i medarbejder tabellen, som så herved kun kan være tilknyttet en bestemt organisation.
b. Projekter deles op i 2 tabeller, sager og projekter, hvor alle arbejdsmuligheder skal indgå i en sag. Hvis der er tale om et projekt tilføjes også data til projekt tabellen, da en sag også kan være et projekt.

Desuden er der foretaget små rettelser på sekvensdiagrammet for use casen efter implementation og tilføjet nogle ekstra ting til den oprindelige use case beskrivelse.

Evaluering over 2 iteration:

I denne iteration blev use casene UC 4, UC 5 og UC 14 implementeret. Her blev det fundet nødvendigt at checke på helligedage ved tildeling, samt på skudår ved alle 3, da arbejdsplanen ellers ville blive forkert her. Desuden blev use casen ”rediger medarbejders arbejdsplan” delt op i 2 nye use case UC 4 og UC 14, da det var mere hensigtsmæssigt at have dem i 2.

Evaluering over 3 iteration:

I denne iteration blev use casene UC 15 og UC 9 implementeret. Her vil månedens time kapacitet blive tilskrevet værdien 0, hvis der ikke findes nogen normtider for måneden. Derfor vil værdier af beregningerne af månedens belastning, forventet uddebitering, og sikre uddebitering få tilskrevet værdien 0 % for at undgå at lave en division med 0 i nævner.

Evaluering over 4 iteration:

I denne iteration blev use casene UC 10, UC 11, UC 1 og UC 8 implementeret.
Her blev det besluttet, at arbejdsplan for en medarbejder(UC 1) også skulle laves, så den kunne vise på månedsbasis for medarbejderen i stedet for på ugeplan. Desuden blev det fundet nødvendigt at bruge en transaktion ved indsættelsen af data for en medarbejder(UC 11) og et projekt(UC 10), da indsættelse af data for disse foregår i flere tabeller. Dette blev gjort, fordi databasen ikke skulle indeholde mangelfulde data for en medarbejder eller et projekt. UC 8 blev tilføjet i webapplikationen.

Evaluering over 5 iteration:

I denne iteration blev use casene UC 16, UC 7 og UC 6 implementeret.
Her blev de implementerede use case UC 7 og UC 6 tilføjet i web applikation.

Bilag C

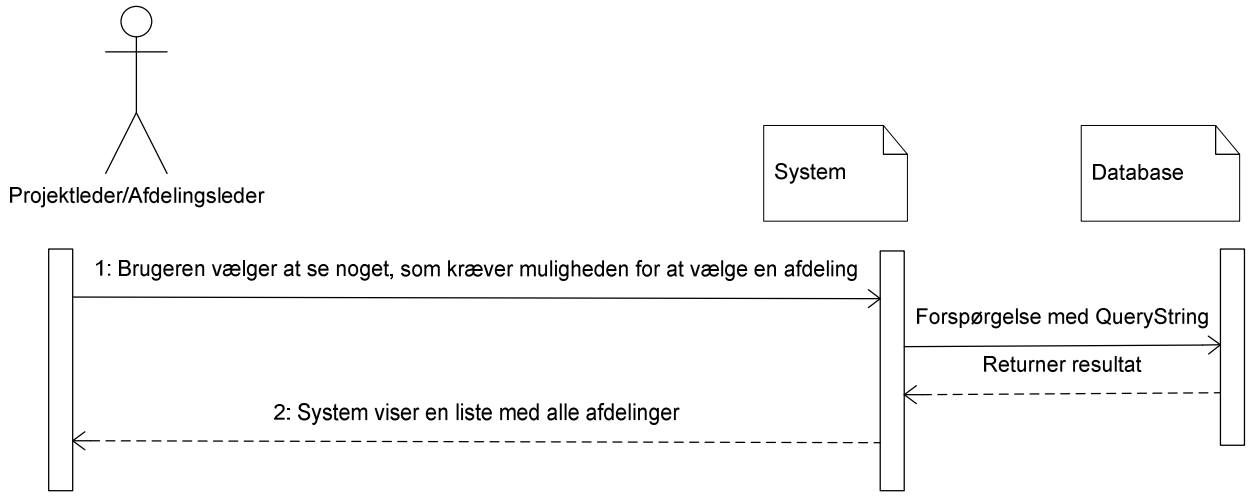
UC 8 Find afdelinger

Use case ID	UC 8	**Use case navn**	Find Afdelinger.
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	**10. August 2007**	**Sidste opdaterings dato**	**10. september**
Beskrivelse af mål med use case	Målet med denne use case er at finde alle afdelinger. Er beregnet til brug sammen med U2. Den bruges inaktivt af brugeren ved at give en liste med afdelinger, som bruger kan vælge imellem.		
Forudsætning			
Succes kondition	Man får en liste med alle afdelinger, som der findes.		
Fejl kondition	Der kunne ikke findes nogen afdelinger.		
Primær Aktør	Projektleder og Afdelingsleder/chefer.		
Sekundær aktør	Medarbejder.		
Trigger	Der er brug for at finde alle afdelinger.		

Normalt forløb	1. Brugeren vælger at se noget, hvor der er brug for at få muligheden for at vælge en afdeling fra en liste med alle afdelinger i organisationen. 2. Systemet viser en liste med alle afdelinger i organisationen, hvis ingen findes, vises en tom liste.	
Alternativt forløb	**Step**	**Branching Action**

Beslægtet information	
Prioritet	5 ud af 100 ikke så vigtig.
Udførelse	Søgning skal foretages hurtigt
Hyppighed	Dagligt
Brugerens brug	Brugeren bruger denne use case inaktivt
Åbne spørgsmål	
Afslutnings dato	14. september 2007
Indgår	UC 2

System sekvens diagram



Figur 56 – System sekvens diagram UC 8

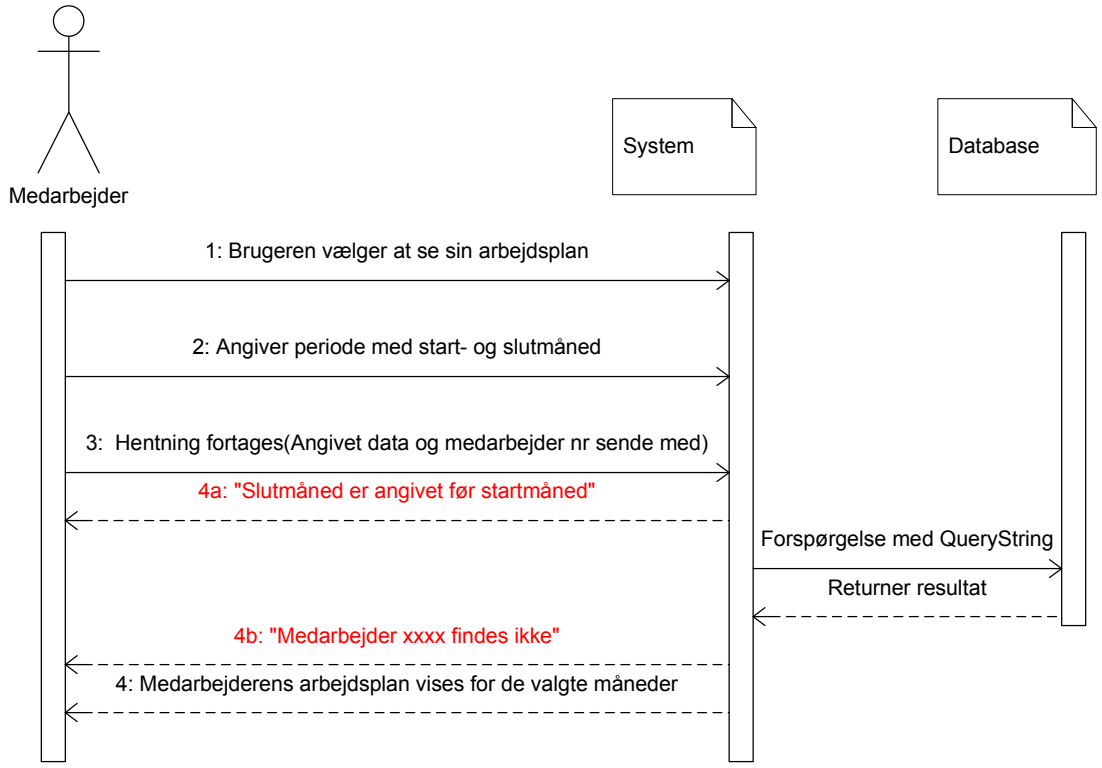
UC 1 Vis min arbejdes måned /Måneder

Use case ID	UC 1	**Use case navn**	Vis min arbejdsplan
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	09. August 2007	**Sidste opdaterings dato**	11. september 2007
Beskrivelse af mål med use case	Målet med denne use case er at give en medarbejder muligheden for at se dennes arbejdsplan for en måned eller flere måneder frem i tiden. Dog max 1 år.		
Forudsætninger	Medarbejderen er identificeret af systemet, så medarbejder nr er kendt		
Succes kondition	Medarbejder får et overblik over de projekter/sager denne skal arbejde med i den valgte periode fordelt på formiddag/eftermiddag		
Fejl kondition	Medarbejderen får ikke at se, hvad denne skal arbejde med i den valgte periode.		
Primær Aktør	Medarbejder		
Trigger	Medarbejderen ønsker at se sin arbejdsplan for en måned eller flere måneder.		
Normalt forløb	1. Brugeren vælger at se sin arbejdsplan. 2. Brugeren angiver en periode med startmåned og slutmåned, som denne ønsker at se arbejdsplanen for. 3. Brugeren vælger at hente arbejdsplan, så data om perioden sendes sammen med medarbejderens nr. 4. System returnerer en arbejdsplan for medarbejderen med informationer om projekter/sager, der arbejdes på i den valgte periode fordelt på formiddag og eftermiddag.		
Alternativt forløb	**Step**	**Branching Action**	
	4a	System viser meddelelsen ”Slutmåned er angivet før startmåned”, hvis der skulle være angivet et forkert interval med månederne.	
	4b	System viser meddelelsen ”Medarbejder xxxx findes ikke”, hvis	

		medarbejderen med medarbejder nr xxxx ikke findes i systemet.

Beslægtet information	Viser den samme arbejdsplan som UC2, dog kun for en medarbejder.
Prioritet	8 ud af 100 ikke så vigtig.
Udførelse	Max 1 min.
Hyppighed	Bruges af og til.
Åbne spørgsmål	
Afslutnings dato	14. september 2007

System sekvens diagram



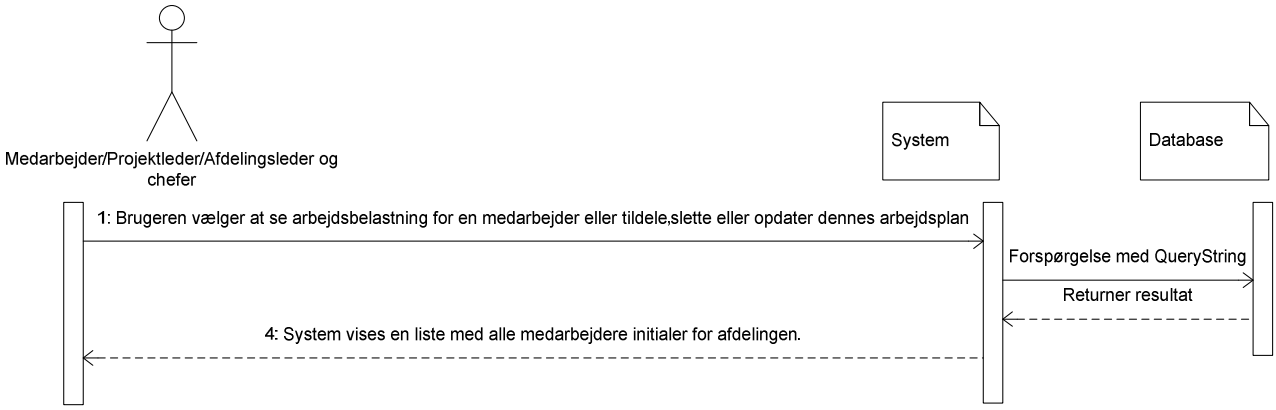
Figur 57 – System sekvens diagram UC 1

UC 6 Find medarbejder / medarbejdere

Use case ID	UC 6	**Use case navn**	Find medarbejder/medarbejdere
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	20. september 2007
Beskrivelse af mål med use case	Målet med denne use case er at finde en medarbejder med relevante data for denne eller alle medarbejdere i en bestemt afdeling med relevante data for disse. Data er i dette projekt valgt til at være medarbejderens initialer.		
Forudsætninger	En afdeling er valgt. I dette projekt køres testwebapplikation med afdelingen ”GIS & IT” valgt.		
Succes kondition	En liste med alle medarbejdere initialer for en afdeling vises.		
Fejl kondition	Der kunne ikke findes nogen medarbejdere.		
Primær Aktør	Medarbejder, Projektleder og afdelingsleder/chefer.		
Trigger	Der ønskes at finde en bestemt medarbejder eller alle medarbejdere for en afdeling.		
Normalt forløb	1. Brugeren vælger at se arbejdsbelastningen for en medarbejder eller at tildele, slette eller opdatere en medarbejders arbejdsplan. 2. System viser en liste med alle medarbejderes initialer for afdelingen.		

Prioritet	5 ud af 100
Udførelse	Søgningen skal gøres hurtigt.
Hyppighed	Benyttes dagligt
Brugerens brug	Brugeren bruger denne use case inaktivt
Åbne spørgsmål	Use casen bruges i dette projekt til at give brugeren mulighed for at vælge en medarbejder med dennes initialer i testwebapplikation. På nuværende tidspunkt bruges den kun til at finde alle medarbejdere i en afdeling.
Afslutnings dato	21. september 2007
Indgår	UC4, UC5, UC 14, UC 9

System sekvens diagram



Figur 58 - System sekvens diagram UC 6

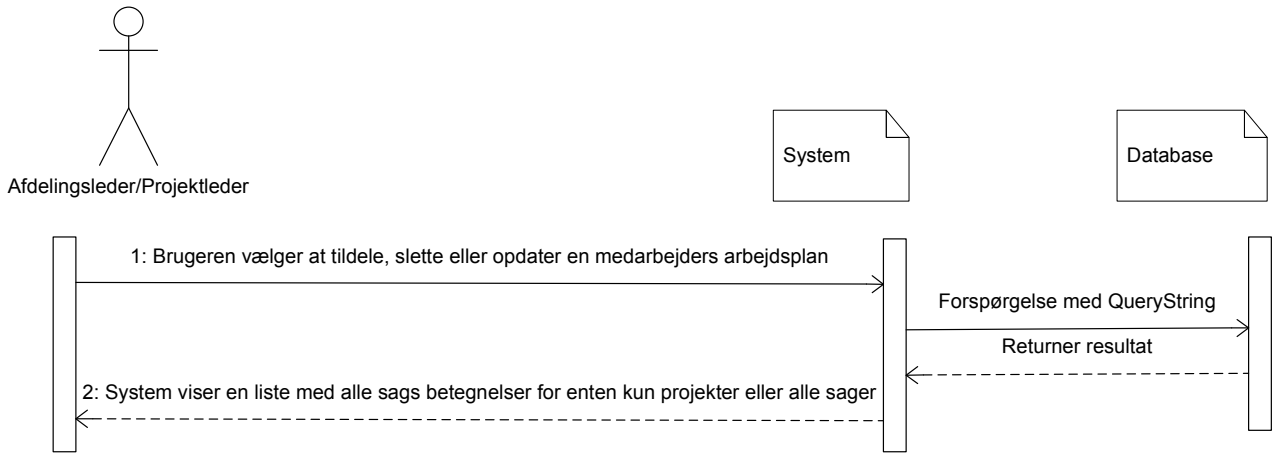
UC 7 Find Projekter / Sager

Use case ID	UC 7	**Use case navn**	Find Projekter/sager
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	17. september 2007
Beskrivelse af mål med use case	Målet med denne use case er at finde alle sager eller alle projekter med tilhørende oplysninger, så brugeren har mulighed for at vælge en bestemt sag eller projekt, som findes i systemet.		
Forudsætninger			
Succes kondition	Alle projekter eller alle sager findes og sagsbetegnelser vises.		
Fejl kondition	Der kunne ikke findes nogen sager eller projekter.		
Primær Aktør	Afdelingsleder/chefer og projektleder.		
Trigger	Brugeren har brug for at få sagsbetegnelser for alle projekter eller sager.		
Normalt forløb find sager	1. Brugeren vælger at tildele, slette eller opdatere en medarbejders arbejdsplan, hvor der er brug for at få muligheden for at vælge en sag eller et projekt fra en liste med sagsbetegnelser. 2. Systemet viser en liste med sagsbetegnelser for alle sager eller kun sagsbetegnelser for alle projekter.		
Alternativt	**Step**	**Branching Action**	

forløb		

Beslægtet information	
Prioritet	5 ud af 100
Udførelse	Søgning skal ske hurtigt.
Hyppighed	Regelmæssigt
Brugerens brug	Brugeren bruger denne use case inaktivt
Åbne spørgsmål	Use casen bruges ved dette projekt i webapplikation under tildeling, sletning eller opdatering af en medarbejders arbejdsplan, således at den finder alle sager med sags betegnelser. Den giver brugeren muligheden for at vælge en sag med sags betegnelse fra en liste, som skal tildeles, slettes eller opdateres på en medarbejders arbejdsplan. Den bruger på nuværende tidspunkt ikke mulighed for kun, at finde projekter.
Afslutnings dato	21. september 2007
Indgår	UC4, UC14, UC5

System sekvens diagram



Figur 59 - System sekvens diagram UC 7

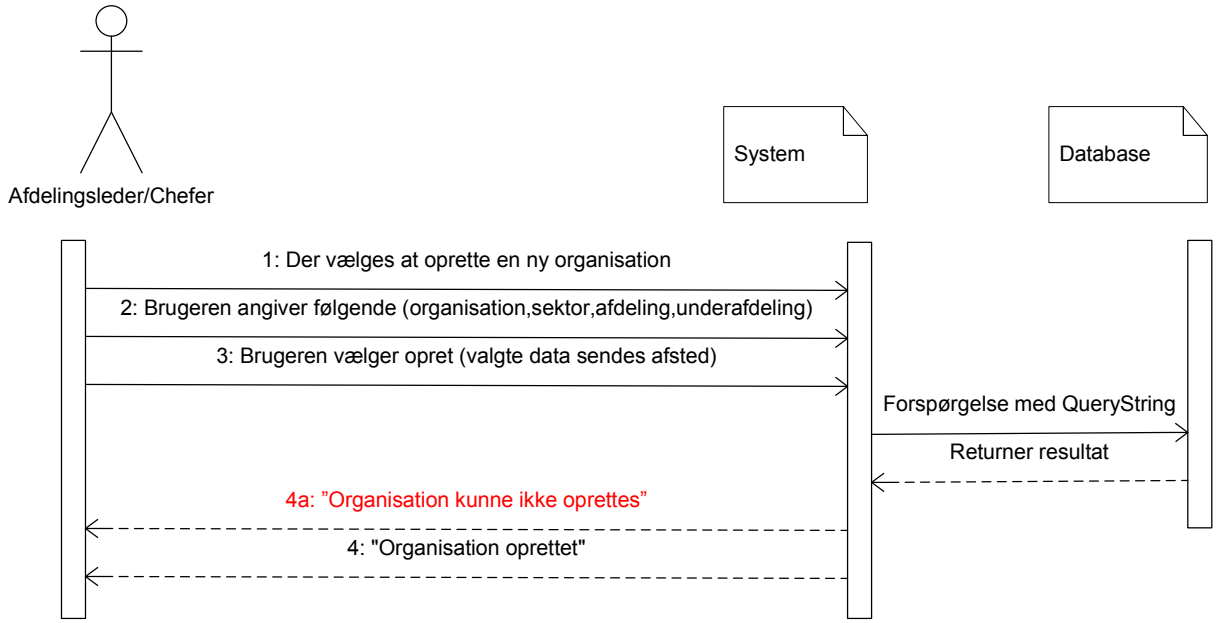
UC 16 Opret organisation

Use case ID	UC 16	**Use case navn**	Opret organisation
Oprettet af	STE	**Sidst opdateret af**	STE
Oprettelses dato	10. August 2007	**Sidste opdaterings dato**	17. september 2007
Beskrivelse af mål med use case	Mål med denne use case er at få oprettet en afdeling eller underafdeling for en sektor i en organisation.		
Forudsætninger			
Succes kondition	En organisationsdel blev oprettet		
Fejl kondition	Organisation kunne ikke oprettes		
Primær aktør	Afdelingsleder/Chef		
Trigger	En organisationsdel ønskes oprettet.		
Normalt forløb	1. Der vælges at oprette en ny organisation. 2. Brugeren angiver følgende. • Navnet på organisation • Navnet på en sektor i organisation • Navnet på en afdeling i sektoren. • Navnet på en underafdeling i en afdeling. 3. Bruger vælger opret organisation 4. System returnerer meddelelsen ”Organisation oprettet”, hvis opretningen lykkes.		
	Step	**Branching Action**	
	4a	System returnerer fejlmeddelelsen ”Organisation kunne ikke oprettes”, hvis noget går galt ved indsættelsen i databasen.	

Beslægtet information	
Prioritet:	5 ud af 100
Udførelse	Oprettelse skal max tage 1 min.
Hyppighed	Hver gang organisation ændres i virksomheden, såsom når nye afdelinger/ sektorer oprettes.

Åbne spørgsmål	Oprettelsen af organisationen bliver i dette projekt ikke lavet I web delen. Men bliver dog lavet som en web services metode. Ikke sikkert der skal være en underafdeling, men gør ikke noget, den er der nu.
Afslutnings dato	14. september 2007

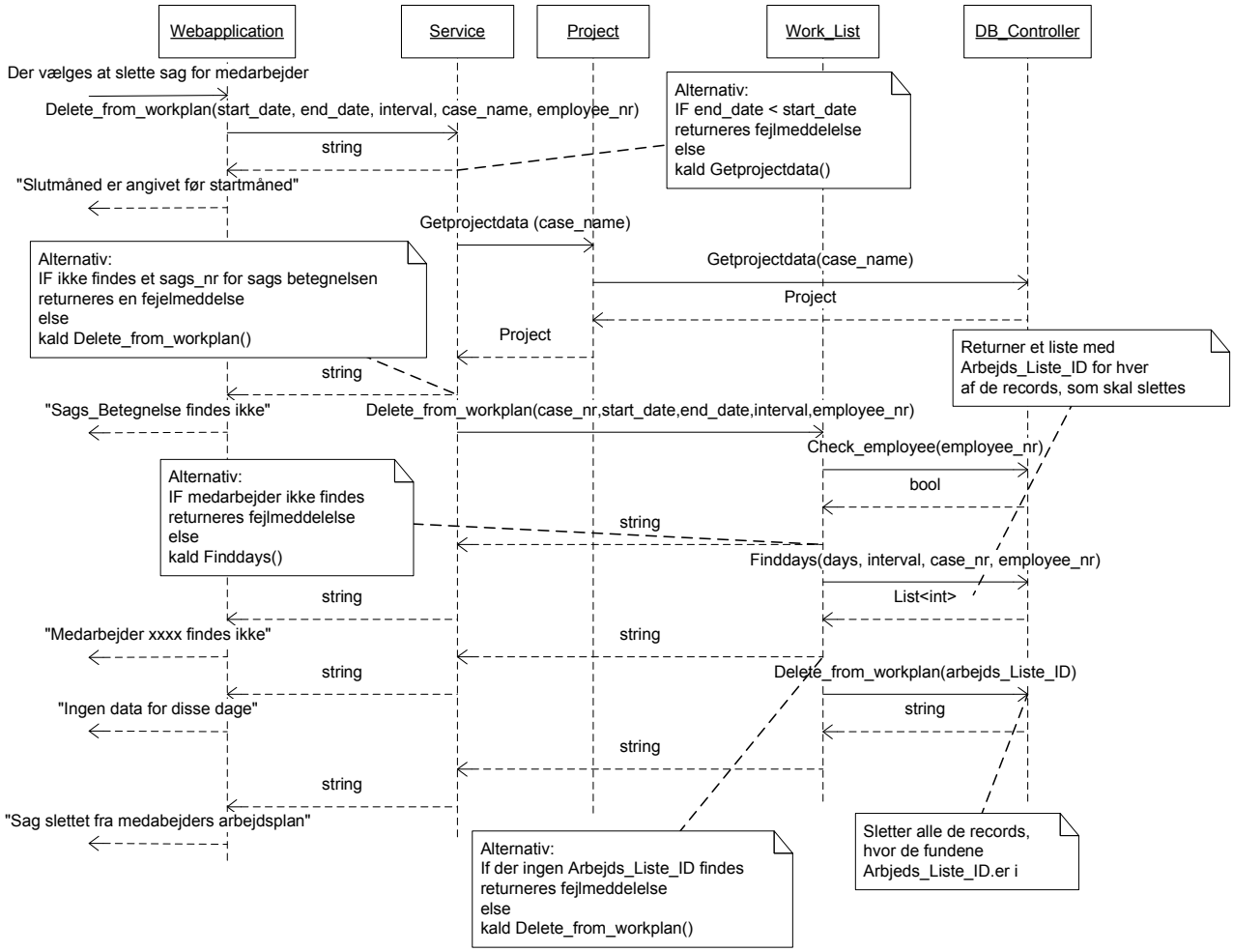
System sekvens diagram



Figur 60 - System sekvens diagram UC 16

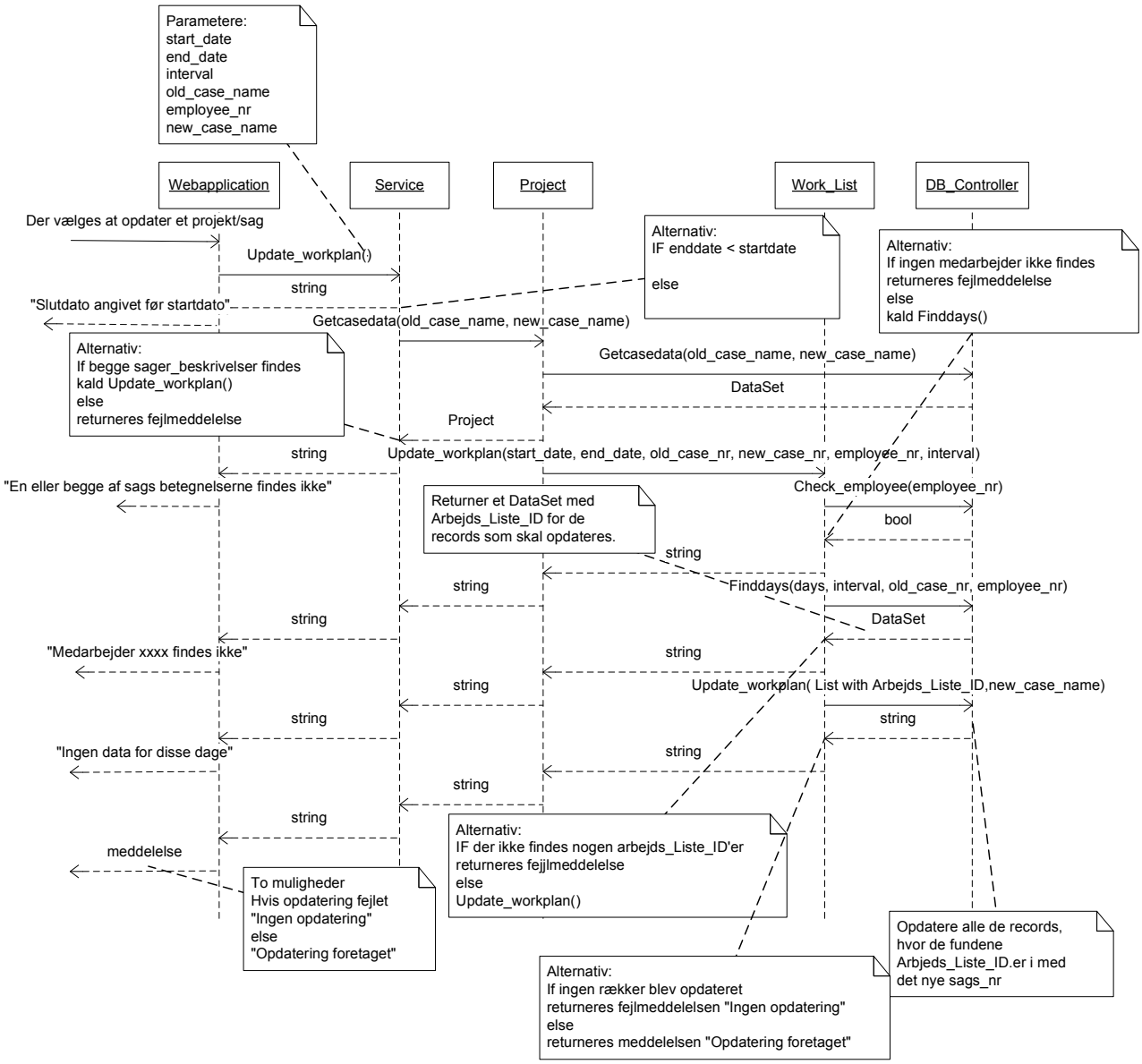
Bilag D

Sekvens diagram UC 4



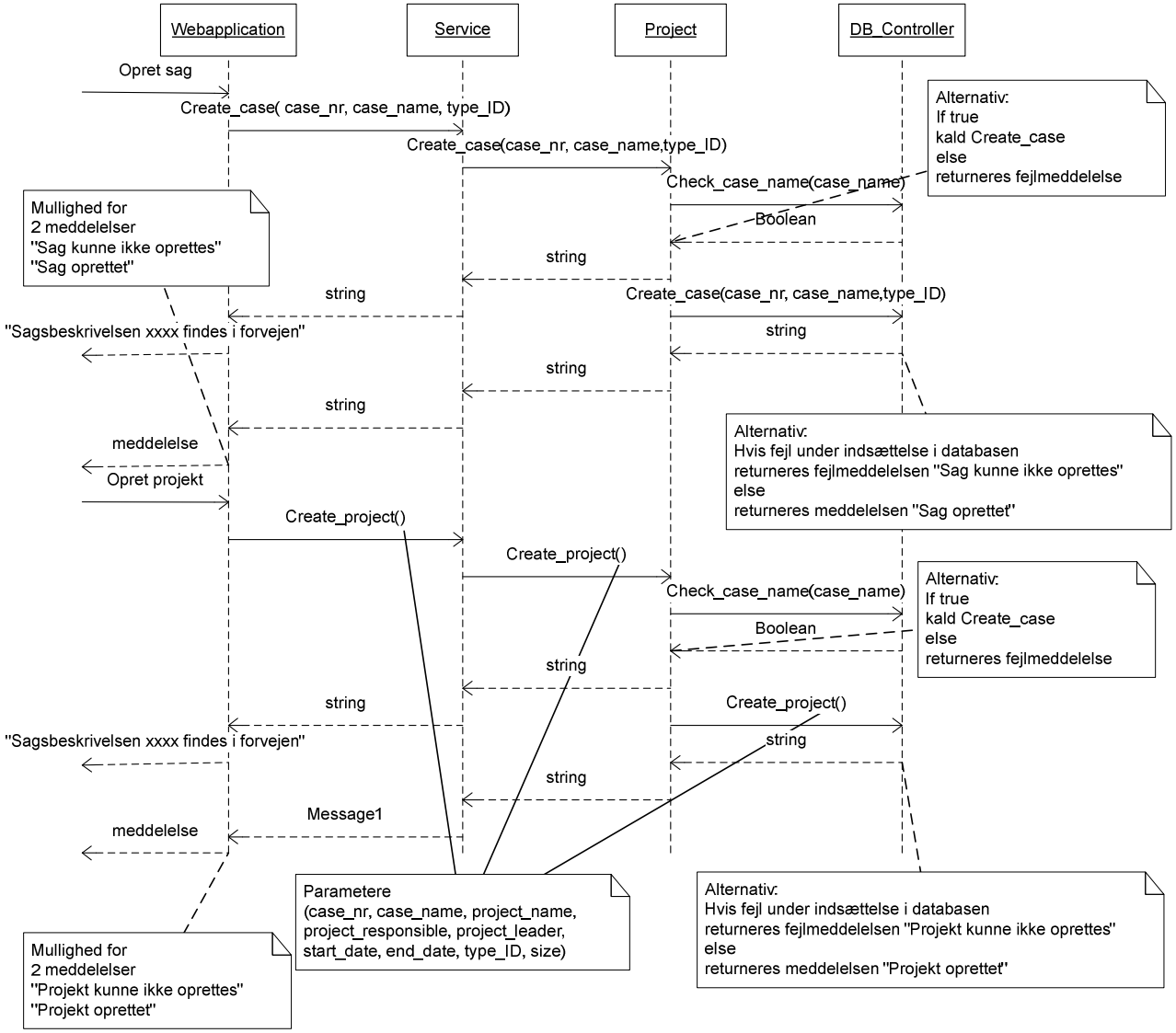
Figur 61 Sekvens diagram for UC 4 Slet projekt eller sag på en medarbejders arbejdsplan

Sekvens diagram UC 14



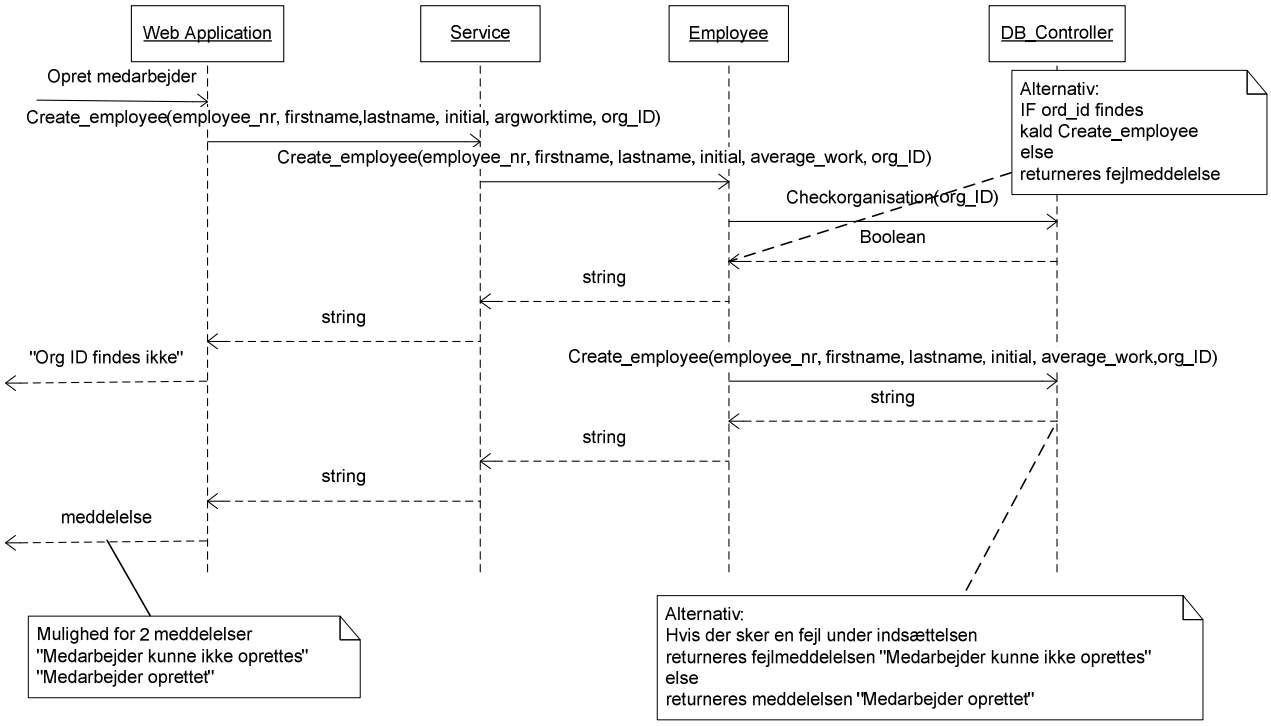
Figur 62 Sekvens diagram for UC 14 Opdater en sag/projekt på en medarbejder arbejdsplan

Sekvens diagram for UC 10



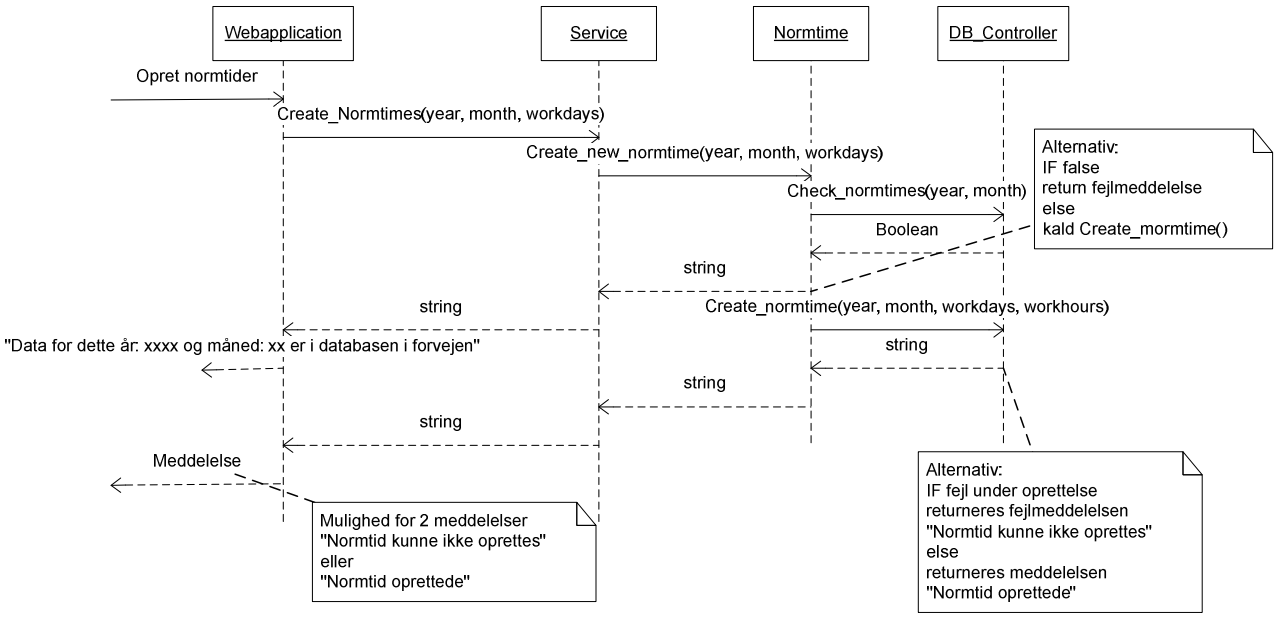
Figur 63 Sekvens diagram UC 10 Opret projekt/sag

Sekvens diagram for UC 11



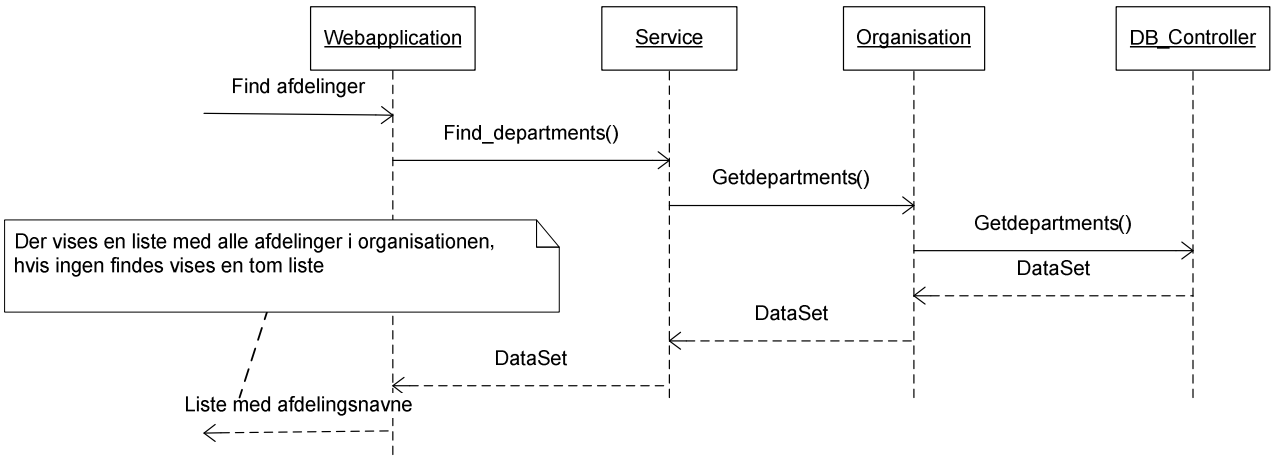
Figur 64 Sekvens diagram UC 11 Opret medarbejder

Sekvens diagram for UC 15



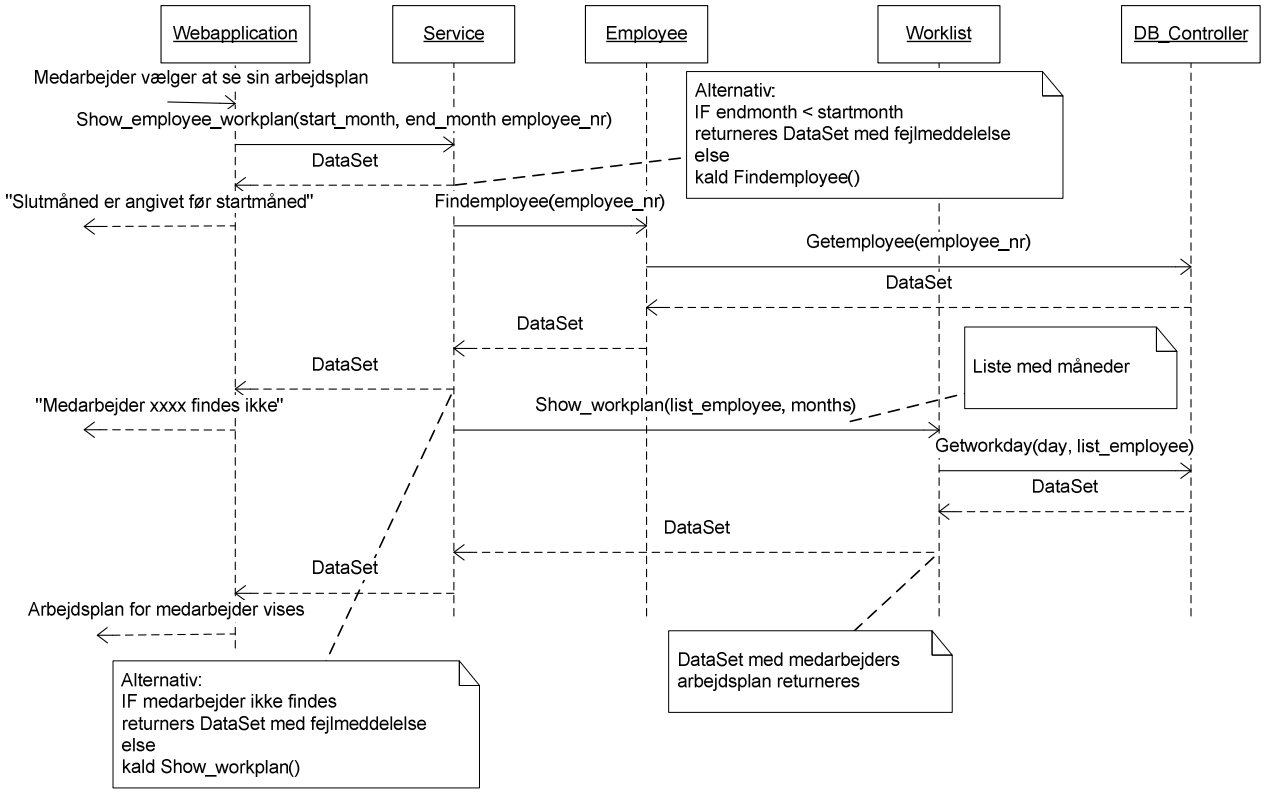
Figur 65 Sekvens diagram for UC 15 Opret normtider

Sekvens diagram for UC 8



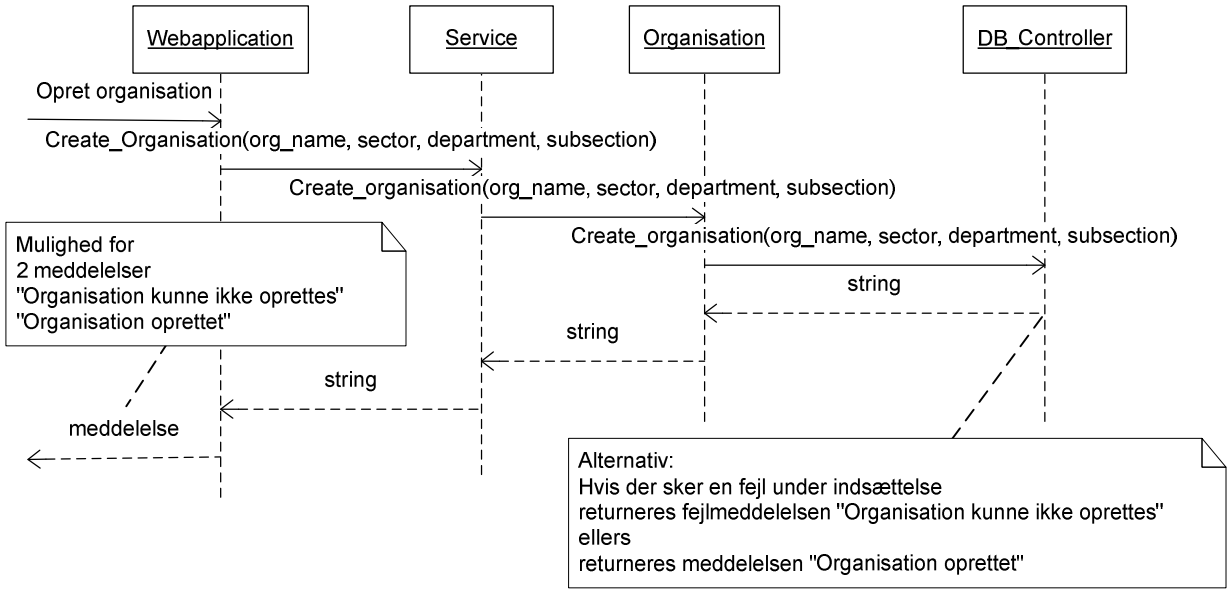
Figur 66 Sekvens diagram for UC 8 Find Afdelinger

Sekvens diagram UC 1



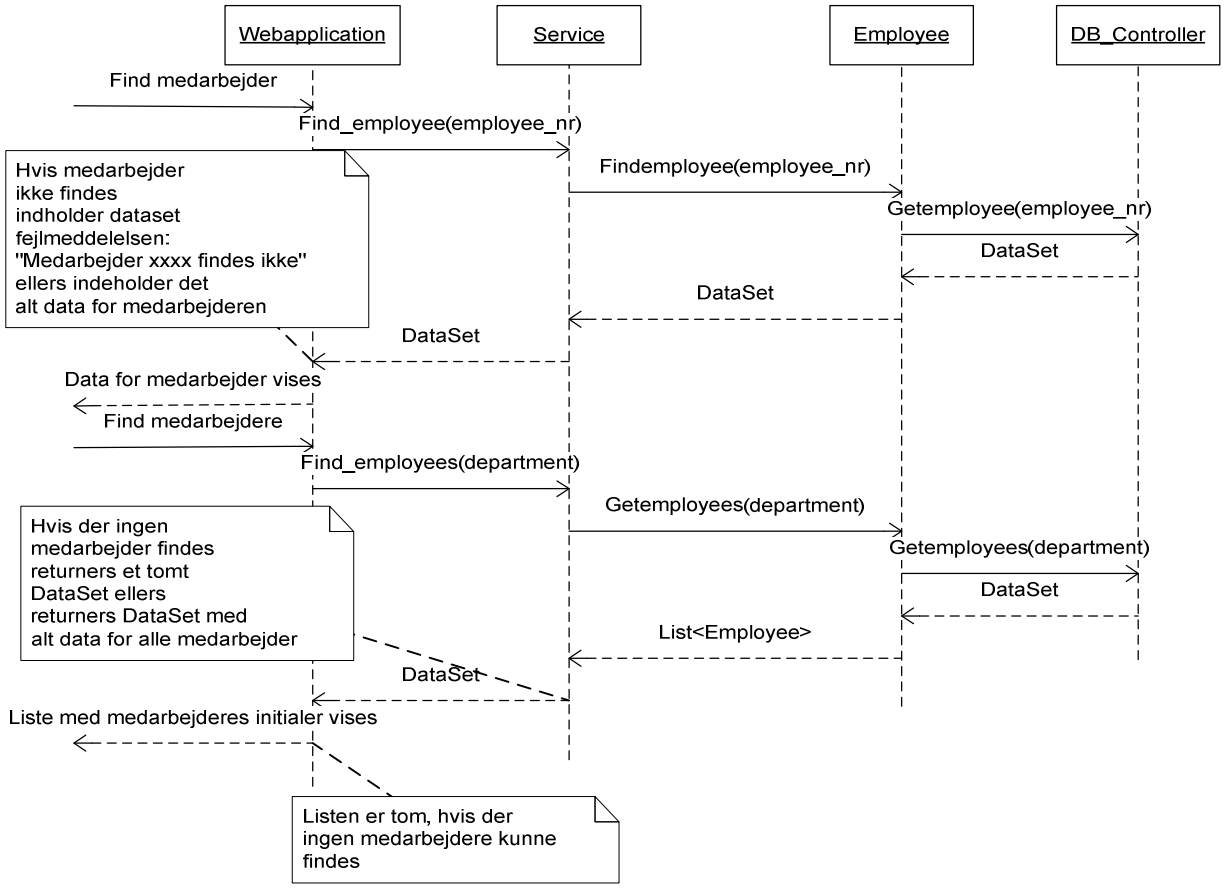
Figur 67 Sekvens diagram UC 1 Vis min arbejdsplan

Sekvens diagram UC 16



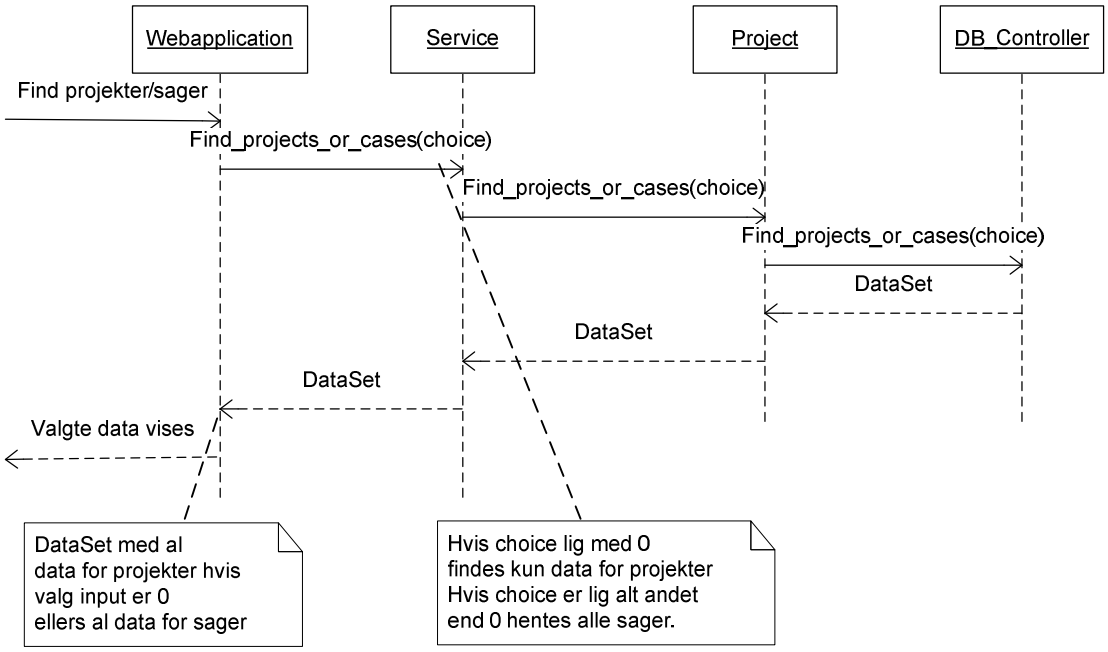
Figur 68 Sekvens diagram UC 16 Opret organisation

Sekvens diagram UC 6



Figur 69 Sekvens diagram UC 6 Find medarbejder / medarbejdere

Sekvens diagram UC 7



Figur 70 Sekvens diagram UC 7 Find projekter/ sager