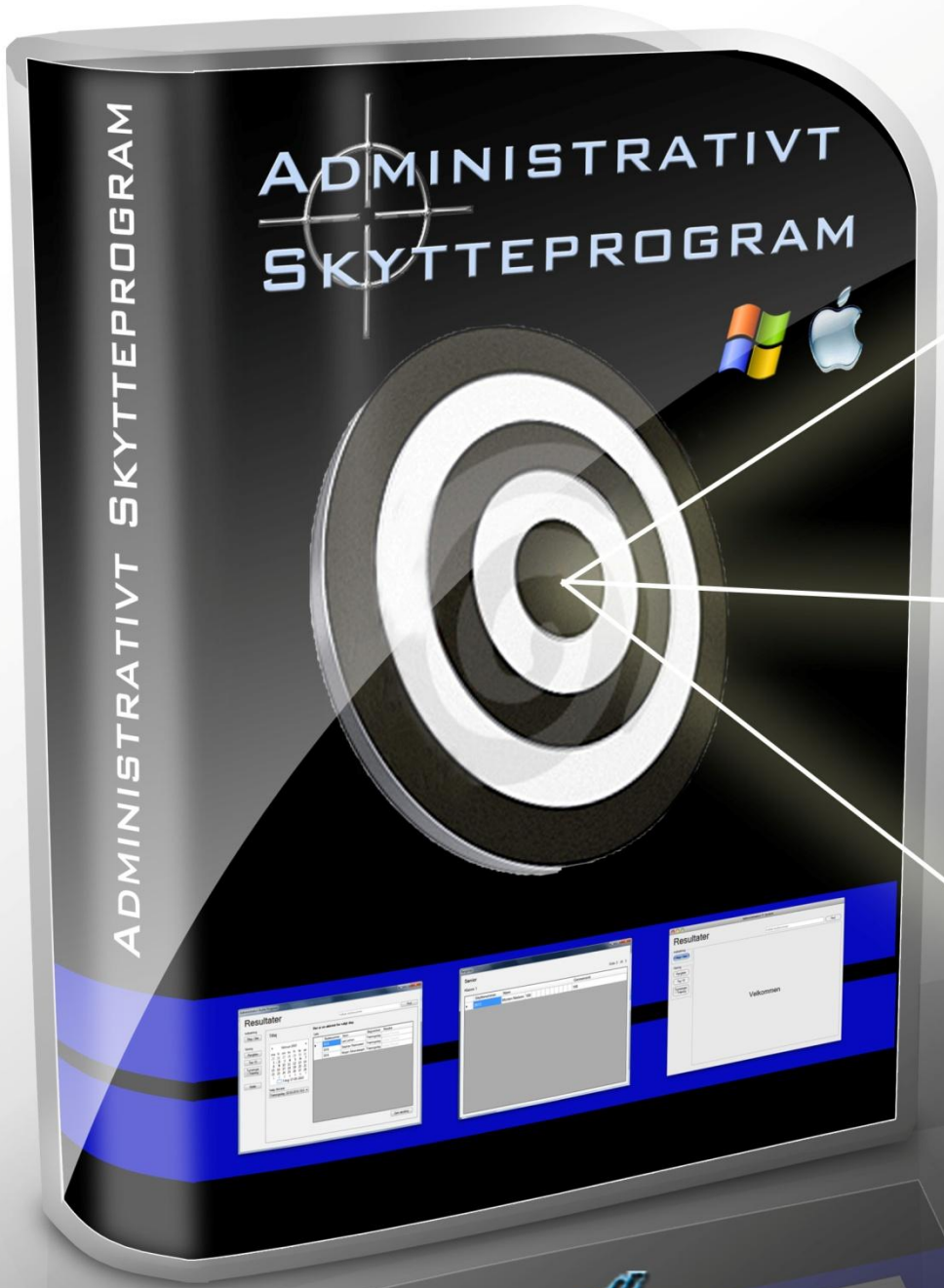
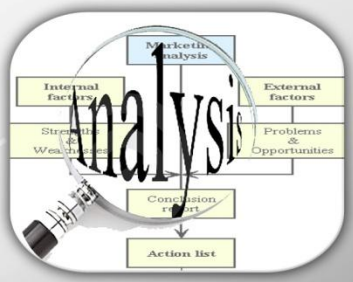
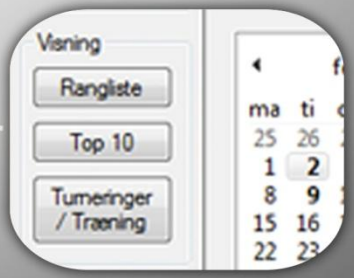
Administrativt Skyttesystem







AALBORG UNIVERSITET

Gruppe B230
Informatik og BAIT

Vejleder: Jesper Kjeldskov
Kristian Albeck, Lars Lichon, Morten Nielsen ,
Nirojan Srikandarajah, Pernille Nielsen, Stephan Rasmussen

P2 Projekt Maj 2010

Titelblad

Titel: Administrativt skyttesystem
Tema: Udvikling af software
Projektperiode: 1/2/2010-27/5/2010
Afleveret: 26/5/2010

Synopsis

Denne rapport omhandler en udarbejdelse af et administrativt resultatsystem til en skytteklub. Igennem rapporten vil processen blive beskrevet via tre caseforløb, som vi er blevet undervist i, i løbet af dette semester.
Vi vil benytte en række relevante modeller og metoder, som har medvirket til fremstillingen af produktet.
Slutteligt vil vi sammendrage disse cases og belyse det som en helhed. Denne sammenslutning ligger til grund for konklusionen på problemformuleringen, hvor vi ligeledes vil vurdere resultatet.

Gruppe: B230

Udarbejdet: Lars Lichon _____

Morten Nielsen _____

Pernille Nielsen _____

Stephan Rasmussen _____

Nirojan Srikandarajah _____

Kristian Albeck _____

Vejleder: Jesper Kjeldskov
Sidetal: 81
Antal bilag: 5
Afsluttet: 26/5/2010
Linjer kode: 3245

Forord

Projektet omhandler systemudvikling til en skytteforening til gavn for de administrative opgaver, herunder resultathåndtering, banekontrol og medlemsstyring. Der har været mange projektforslag på bordet ved udvælgelsen, hvorved skydesporten blev fundet mest interessant. Kontakten til skytteklubben har været særdeles tilfredsstillende for begge parter, hvorved det var muligt at få hjælp til skytteregler og afprøvning af vores endelige prototype.
Vi har i projektforløbet modtaget undervisning i de cases som indgår i denne rapport.
De tre cases som bliver omtalt og forkortet i denne rapport er:

- SAD - Systemanalyse og design.
- DIEB - Design, implementering og evaluering af brugergrænseflader.
- OOPA - Objektorienteret programmering og algoritmik.

Der vil i rapporten fremgå fagtekniske termer, hvilket især finder sted indenfor de tre cases, heriblandt bliver termen ”prototype” benyttet for det udviklede program til skytteklubben. Når termen ”systemet” bliver benyttet i rapporten, menes der hele det analyserede system.
Tidsforløbet for udarbejdelsen af rapporten: d. 1/2-2010 til 27/5-2010.
Vi vil gerne takke vores undervisere, som har hjulpet meget undervejs i forløbet og i særdeleshed Jesper Kjeldskov som vores vejleder.
Vi vil også gerne takke vores testperson fra skydeklubben for hjælp til usabilitytesten.

Det programmerede program er vedlagt på en CD-ROM, sammen med kode dokumentation.

Læsevejledning

Følgende læsevejledning kan give læseren hjælp til at forstå opbygningen af denne rapport.
Rapporten skal læses fra toppen til bunden. Den er opdelt i hovedafsnit med op til i alt 4 underafsnit. Disse underafsnit er identificeret med tal, således det gøres synligt, hvilket hovedafsnit de tilhører. Da en stor del af rapporten bygger på de tre fag SAD, DIEB og OOPA, har hvert af disse fået deres egne afsnit i rapporten, med titlerne ”Analyse og design”, ”Brugergrænseflade” og ”Implementering”.
Der vil igennem rapporten løbende blive refereret til de bøger, hjemmesider og andre kilder, vi har benyttet os af. Dette vil foregå ved, at hver reference har eget referencenummer, der så kan slås op i referencelisten til sidst i rapporten. Referencer skrives eksempelvist som følgende: [SAD1].

Indholdsfortegnelse

1 Indledning

Dette P2 projekt handler om udviklingen af et administrativt IT-system til en selvvalgt idrætsforening, i dette tilfælde er der tale om et administrativt IT-system til brug i en skydeklub. Systemet ligger vægt på tre elementer, hvilket er som følgende:

- Resultathåndtering indebærer visning af resultater til klubbens medlemmer, samt kontrol og styring af resultater som klubbens medlemmer har skudt.
- Medlemskontrol indebærer kontrol over klubbens medlemmer, heriblandt at tilmeld medlemmer til klubben og have kontrol over skytternes placeringer i grupper og klasser.
- Banebookingsdelen af systemet gør det muligt, at reservere baner. På denne måde sikrer skytten sig, at der er en bane til ham/hende på det tidspunkt skytten har lyst til at skyde.

Alle tre dele af systemet arbejder sammen på en logisk måde, blandt andet vil ændringer af resultater give udfald i den del af programmet der håndterer medlemmer, eksempelvis ved at fortælle aktøren af systemet at en handling skal foretages.
Formålet med dette P2-projekt er, at kunne analysere, designe, programmere og evaluere et IT-system, samt danne viden indenfor grundlæggende datalogiske emnekredse og benytte teknikker til udvikling af software. Endvidere har projektet til formål at danne viden om problem- og projektorienteret arbejde.

For at kunne programmere et brugbart system, er der blevet undervist i forskellige kurser, som har gjort den stillede opgave muligt at løse. Til analysen og designet af systemet, er der blevet brugt de metoder og beskrivelsesteknikker, som er blevet dækket i kurset Systemanalyse og Design (SAD). Systemet skulle udvikles objekt-orienteret, og til dette er der blevet undervist i Objekt-orienteret Programmering og Algoritmik (OOPA), hvor C# har været programmeringssproget. I kurset Design, Implementering og Evaluering af Brugergrænseflader (DIEB), blev der undervist i teknikker, som er blevet brugt til at designe og evaluere systemets brugergrænseflade.

1.1 Problemformulering

Temaet for dette P2 projekt er udviklingen af et IT-system:

- Hvilke processer er der under udviklingen af et administrativt it system, og hvordan skal dette system opbygges så det dækker skytteklubbens behov på en brugervenlig måde?

Der analyseres, designes, programmeres og evalueres et administrativt system i denne rapport, ved brug af kompetencerne opnået i de tre kurser. Det færdige administrative system evalueres ved hjælp af en usability test, en black box test og en IDA session.

1.2 Metode

Rapportens opbygning er baseret på henholdsvis de tre case forløb, SAD, DIEB og OOPA. Hvert casemodul er bestående af en selvstændig indledning, metode og analyse afsnit samt en delkonklusion.
De tre case moduler vil blive indkapslet i samme struktur. Væsentlige udfordringer vil blive diskuteret i et diskussionsafsnit inden konklusionen. Udarbejdelsen af Aalborg Skyttekreds administrationssystem vil ske med programmet Microsoft Visual C# 2008 Express Edition.
Tidligt i arbejdsprocessen er der aflagt besøg hos Aalborg Skyttekreds, for at få indblik i de administrative opgaver, regler samt skyttemæssige forudsætninger.
SAD casemodulet, kapitel 2, beskriver hvilke tanker og overvejelser, der ligger forud for systemet. Denne del af rapporten vil bære præg af metoder og modeller som eksempelvis BATOFF, hændelsestabel og tilstandsdiagram. Her vil systemkrav blive beskrevet detaljeret, klasser, objekter og attributter vil blive defineret med det formål at skabe forståelse for systemets grundstruktur. Systemets funktionalitet vil også blive beskrevet i dette afsnit via en funktionsliste, som vil skildre relevante funktioner.
En prioriteringsliste vil herefter vise vigtigheden af visse kriterier for grænsefladen. Denne vil i særdeleshed være væsentlig for den videre udvikling af systemet. For bedst muligt at danne et overblik over udviklingsprocessen, vil navigationsdiagrammer og vinduesdiagrammer skildre systemets komponentdele. Det er essentielt for systemets udviklingsproces, at disse begreber bliver gennemarbejdet forud for implementeringen, for at sikre korrekt konstruktion.
Administrationssystemet er heri illustreret i sin grundform via en variation af modeller, metoder og diagrammer.
Objektorienteret analyse og design (OOA&D)metoden, som er blevet benyttet i SAD casemodulet, er blevet skræddersyet til dette system, dvs. at nogle modeller og metoder er blevet valgt fra og der er blevet fokuseret mere på andre modeller og metoder. De tre sidste aktiviteter i OOA&D, Processer, Design af kompo-

nenter, Strategi, er blevet valgt fra i dette case modul, da der bliver arbejdet med disse tre aktiviteter i kapitel 3, som omhandler brugergrænsefladedesign.

Casemodulet DIEB, kapitel 3, er en videre gang af det foregående case modul, men hvor SAD vægter systemkriterier og funktionalitet, har DIEB fokus på designet af brugergrænsefladen, analyse og evaluering heraf. Aktørerne udgør her en vigtig faktor, derfor vil disse blive defineret gennem stakeholder analyse og en personas model. Dette vil give en mere dybdegående indsigt i det eksisterende brugerbehov, som er det essentielle for udviklingen af grænsefladen.

Prioriteringstabel etableres, hvori de mest væsentlige kriterier til grænsefladen pointeres efter prioritet. Dette giver en bredere forståelse af, hvordan systemet skal sammensættes og hvilke aspekter, der er relevante at fokusere på. Sideløbende er det relevant at implementere komponenterne således det giver aktøren en nem interaktion med systemet. Til dette anvendes Gestalt lovene, mapping metoder og consistency, disse skal give aktøren en logisk tilgang til systemet. Sidst i casen bliver der lavet en usabilitytest, hvortil IDA metoden vil blive anvendt.

Tredje case er OOPA, kapitel 4, og dette modul vil vægte kodebeskrivelse, dokumentation, systemet og usabilitytest. Udvalgte stykker kode vil blive detaljeret beskrevet, resten af programkoden vil dokumenteres ved hjælp af programmet Doxygen.

I kapitlet vil der indgå væsentlige ændringer under konstruktioner af systemet, argumentationer og en præcis beskrivelse af klasser og metoder vil blive redegjort for.

Afslutningsvis vil systemet blive evalueret og testet på; til systemtesten vil NUnit version 2.5.3 blive anvendt.

En prototype af systemet vil blive konstrueret. Prototypen vil være en lille gren af hele systemet.

1.3 Casebeskrivelse

I denne case er der valgt at systemudvikle et administrationssystem til Aalborg Skyttekreds. Der tages udgangspunkt i klubbens nuværende administrationssystem.

1.3.1 Om klubben

Klubben er beliggende på Skydebanevej 1 i 9000 Aalborg. Klubben råder over 12 indendørs skydebaner på samme adresse. Ydermere har klubben eksterne udendørs skydebaner, som er beliggende i Dall. Der bliver skudt med rifler og pistol i forskellige discipliner, som eksempelvist luft og 15 meter.
Hver skytte, som er tilmeldt klubben har et unikt skyttenummer.

Kontakt til klubben

Kontakten til klubben blev etableret via gruppemedlemmet Lars Lichon, som er medlem i Aalborg Skyttekreds. Gruppens øvrige medlemmer har alle været ude på Aalborg Skyttekreds' baner for at se klubben og skabe kontakt og dialog til klubbens medlemmer. Primær kontaktperson er Lars Ørum, aktiv skytte og frivillig afløser til de administrative opgaver i Aalborg Skyttekreds.

Nuværende administrationssystem

Det nuværende administrative system består af både manuelt arbejde og et IT-system.
De administrative opgaver består i:

- At holde styr på klubbens medlemmer manuelt, Dette indebærer blandt andet at sørge for at diverse klubmedlemmer er i de rigtige klasser og grupper. Samt herudover blandt andet, at sørge for klubbens medlemmers våben registreringer.
- At holde styr på banereservering manuelt, Dette indebærer blandt andet at sørge for at skytter ikke kan skyde på samme baner på samme tid. De administrative opgaver, der har med bane booking at gøre i Aalborg skydeklub er dog i det minimale, da skytter som regel blot finder en tom bane og skyder på denne.
- Manuel indtastning af resultater for efterfølgende printning og ophængning af resultatliste, dette er den eneste måde klubbens medlemmer kan se oversigt over sine egne og andres resultater.

Disse opgaver er alle tidskrævende og omstændelige processer, som kan effektiveres.

1.3.2 Regler

Udarbejdelse af et administrationssystem er en omfattende proces. Det specifikke regelsæt for skydeklubben har betydning for systemudviklingen og det endelige administrationssystem, da det skal kunne vise korrekt resultatvisning.

De officielle regler for skydning er DDS- Regelsættet, herud fra fremgår det at pågældende skytte skal foretage 25 skud i en serie, hvoraf de første 5 er prøveskud og de sidste 20 skud vil være de gældende. Herefter forekommer en sammenlægning af de 20 gældende skud, hvorefter sammenlægningen divideres med antallet af gældende skud, og resultatet for en skydning er dermed fundet (der henvises til bilag 4, for et billede af en skydeskive). En skyttes niveau beregnes ud fra et gennemsnit. Dette gennemsnit er en sammenlægning af maksimalt 10 skydninger og overskrider antallet af skydninger 10, vil de 10 bedste skydninger medgå i gennemsnittet. Skyttens gennemsnit vil sideløbende med alder være en faktor for skyttens placering, i henhold til gruppe og klasse.

Hver skytte er inddelt i en gruppe inddelt efter alder. Hver gruppe har op til fire klasser, som inddeler skyterne efter færdighed, som måles på skyttens gennemsnit. En skytte der er i gruppen ”senior”, klasse 2 er således bedre end en anden skytte i ”senior” klasse 3.

Reglerne er udledt fra Skyttebogen 2009-2010 [SAD2].

2 Analyse og Design

Objektorienteret analyse og design, OOA&D, er en metode, som bruges til at udvikle softwaresystemer. Denne metode indeholder en række modeller, som hjælper til udviklingen af programmet. Modellerne er kendetegnet ved at illustrere klasser, tilstande og opførselen.
Metodens analysedel fokuserer på, hvad systemet kan og designdelen viser hvordan systemet er opbygget.

Igennem et OOA&D forløb finder man fire hovedaktiviteter, som er henholdsvis analyse af problemområdet, analyse af anvendelsesområdet, design af arkitektur og design af komponenter. Når der analyseres gennem disse hovedområder bliver arbejdsprocessen meget iterativ. Dette vil sige, at der ofte vil komme til ændringer i noget, der allerede er arbejdet med, i takt med at ny viden opnås.

Denne objektorienterede tilgang har til formål, at se nærmere på problemløsningen i et projekt. Man analyserer sig frem til den bedste løsning via en række modeller og metoder, der illustrerer dette for at give et bedre overblik.

Først i udviklingsprocessen fastlægges en systemdefinition, hvilket sikrer at alle udviklere er bekendt med systemets grundidé og funktion. Herefter analyseres der videre på problemområdet, som er den del af systemet, som hjælper aktøren i anvendelsesområdet til at løse sine opgaver. Efter endt analyse af problemområdet, fokuseres der på anvendelsesområdet, som er det område der administrerer problemområdet og samtidig har stor fokus på aktørerne.
Slutteligt skal man fokusere på arkitekturen i systemet. Her skal der opnås en forståelse af systemet som et samlet hele.

Formålet med SAD afsnittet er at få en struktureret tilgang til udviklingen af systemet til skydeklubben. Ved at bruge denne objektorienteret tilgang, opnås der en struktur og et overblik til design af systemet. Undervejs i processen hjælper de forskellige diagrammer til bedre forståelse som en naturlig del af processen.
Da der bruges mange diagrammer til at visualisere systemet, giver det også en god forståelse udviklere og kunder imellem.
Til dette kapitel henvises der til bogen Objekt Orienteret Analyse & Design [SAD1]

2.1 Afgrænsning

Der afgrænses fra at beskrive systemgrænsefladen. Dette skyldes, at det ikke er relevant for systemet og der skal ses bort fra dette. Derfor analyseres der heller ikke på web-baserede programmer og database-struktur.

Analysedelen i casen bliver beskrevet mest, og derfor vil designdelen i dette afsnit kun blive belyst overfladisk. Dette skyldes, at der i kapitel 3 vil være meget fokus på brugergrænsefladedesign.

Der afgrænses fra at analysere på udendørs skydning, da det minder om indendørs skydning rent funktionsmæssigt og programmeringsmæssigt.

I reglerne til skydning fokuseres der kun på DDS (De Danske Skytteforeninger). Derfor afskrives der fra reglerne for DSU (Dansk Skytte Union). Dette skyldes, at Aalborg Skyttekreds skyder efter DDS regler.

Der ses heller ikke på hold-skydningen i hverken turnering eller træning, efter vejledning fra kontaktpersonen i skydeklubben. Dette skyldes at reglerne for holdskydning vurderes til at være for komplicerede.

Aktørernes interaktion med systemet er begrænset, således at de kun kan aflæse resultater på en ekstern skærm. De kan derfor ikke selv indtaste noget i systemet, grundet der ikke vil blive designet til touch-screen.

2.2 Metode

Metodeafsnittet af rapporten vil blive struktureret således, at der dannes et overblik over alle de komponenter der skal indgå forud for programmeringsdelen. Til dette benyttes relevante modeller og diagrammer for, at fremme denne struktur.

Først illustreres forskellige situationer via rige billeder, og her beskrives grafisk de forskellige dele i anvendelsesområdet af skydeklubben. Modellen BATOFF anvendes til, at opstille relevante kriterier for systemet, og modellen bruges til at danne systemdefinitionen.

Der bliver gjort overvejelser omkring hvilke klasser, der er at forstille i problemområdet, samt de enkelte hændelser i hver enkel klasse. Klasserne skal være med til at beskrive mængden af objekter i systemet, som indeholder sammenlignelig struktur, adfærdsmønster og attributter. Ved at udforme en hændelsestabel, skildres de enkelte klasser og de hændelser der er tilknyttet. Der opnås herved et overblik over klasserne og de fælles hændelser imellem dem.

Det er vigtigt, at sammensætningen af disse sker på en struktureret måde og derfor laves et klassediagram, hvori det fremgår hvordan og på hvilken måde systemets klasser knytter sig til hinanden.

Et tilstandsdiagram laves for at beskrive de enkelte klasser og hændelser. Der analyseres på de tilstande, hver klasse gennemgår under forskellige situationer. Formålet er, at opnå et overblik over kriterierne for hver enkel klasse forud for programmeringsdelen, samt overgangene klasserne imellem.

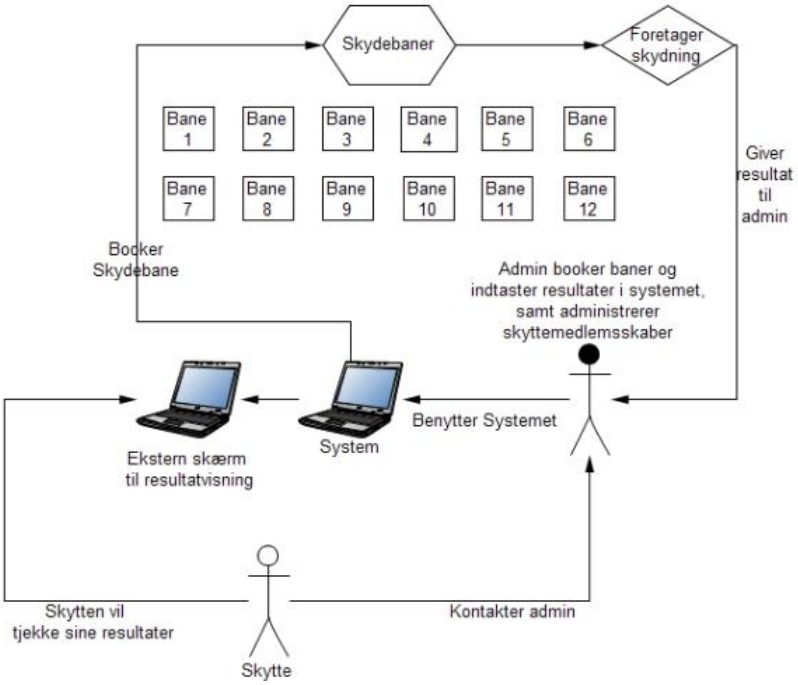
Der kan nu udarbejdes en funktionsliste, som beskriver en komplet oversigt over relevante funktioner, som skal fremgå i systemet. Man kan her vurdere kompleksiteten af de enkelte funktioner, og derfor hvordan de skal repræsenteres i systemet.
Systemets brugergrænseflade kan nu laves ved at vurdere navigationsdiagrammer og vinduesdiagrammer. Disse illustrerer grundstrukturen i systemet, og herefter kan der opstilles kriterier for brugergrænsefladen i form af en prioriteringsliste.
Afslutningsvis ses der på komponentdelene i systemet. Der er her fokus på forståelsen og struktureringen af systemet. Formålet er, at skabe fleksibilitet i systemet, via beskrivelse af de enkelte helheder og systemdele, som kan tage et fælles veldefineret ansvar. Efter råd fra vejlederen er der blevet valgt, at fokusere mindre på denne del af rapporten. Der er derudover blevet valgt at lave et komponentskema og et diagram over komponenterne. Formålet er henholdsvis at prioritere mål for design, for at give en bedre ide til hvad der blandt andet skal være fokus på, og at give oversigt over diverse komponenter i systemet, samt herunder deres interaktion med hinanden og deres ansvarsområde.

2.3 Systemvalg

Her fastlægges der hvordan systemet grundlæggende skal fungere, og hvilke krav det skal opfylde. Først laves der et rigt billede, for at danne et overblik. Derefter bruges der BATOFF-kriterierne, for at fastlægge de indbyggede betingelser. Ud fra dette, konstrueres en systemdefinition, på baggrund af betingelser fra BATOFF-kriterierne. Der henvises til kapitel 2 i bogen Objekt Orienteret Analyse & Design, [SAD1]

2.3.1 Rigt billede

I begyndelsen af et projekt er det vigtigt, at finde frem til hvad det er systemet skal anvendes til. Man laver derfor en række rige billeder for at udvælge det område, hvor der er størst behov for støtte. På Figur 1 ser man en situation, hvor der bliver booket en bane og administreret resultater.



Figur 1 - Rigt billede

2.3.2 BATOFF

BATOFF kriteriet kan anvendes til at støtte udarbejdelsen af en systemdefinition. Modellen bruges til at fastlægge nogle krav til systemet, samt sikring af samme forståelse hos udviklere under programmering og fælles enighed med kunden.

Betingelser

Systemet skal være let og præcist at bruge for administrator, og skal give overskuelige og præcise resultater for administrator og skytter. Systemet skal hjælpe klubben og dets medlemmer til at holde sig opdateret på egne fremskridt, såvel som andres. Herudover skal systemet kunne administrere banefordeling, både til træning og turnering. Systemet skal også kunne håndtere medlemsinformationer. Ydermere skal det være enkelt og brugervenligt. Systemet skal også kunne bruges til tilmelding og afmelding af både klubben og turneringer.

Anvendelsesområde

Systemet skal kunne anvendes direkte af en administrator, samt indirekte af skytter, med viden indenfor skydning.

Teknologi

Systemet skal have en ekstern skærm, der kan benyttes af klubbens medlemmer, til at holde sig opdateret på egne og andres fremskridt. Systemet skal udvikles i C# i programmet ”Microsoft Visual C# 2008 Express Edition”. Systemet skal som minimum kunne operere på operativsystemet Microsoft Windows XP SP3, eller nyere.

Objekter

Bane, Tid, Gruppe, Resultat, Discipliner, Turnering, Træning, Skytte.

Funktioner

Systemet skal benyttes til at administrere resultat fordeling i forbindelse med turneringer og træning, samt disse resultater skal være tilgængelige for skytter. Systemet skal ligeledes også indeholde en booking funktion, samt afmelding og tilmelding til både klub og turnering.

Filosofi

Administrativt og oplysnings værktøj til gavn for skytte og forening.

2.3.3 Systemdefinition

Systemet skal bruges til arbejde med banebooking, til henholdsvis skydetræning og konkurrenceskydning, samt tilmelding og afmelding til både klub og turnering/træning. Systemet skal kunne registrere resultater over indendørs skydning på 15m (meter), og luft i forhold til diverse gruppefordelinger. Endvidere skal systemet primært anvendes til administrativt arbejde, men samtidigt give deltager(e) en mulighed for at følge med i den enkelte skyttes samt egne fremskridt, via en ekstern skærm.

Systemet skal baseres på en PC med de gængse IT værktøjer. Systemet skal kunne opereres af brugere med forskellige erfaringsniveauer indenfor IT. Systemet skal kunne køre på en kontor pc med operativsystemet Microsoft Windows XP SP3 eller nyere.

2.4 Analyse af problemområde

I analysen af problemområdet, fokuseres der på hvilken information systemet skal behandle. Tre aktiviteter analyseres, hvor der udvælges de objekter, klasser og hændelser, som skal udgøre byggestenen i modellen af problemområdet. Denne aktivitet kaldes for klasseaktiviteten og dernæst skal modellen sammensættes, ved at der ses på de strukturelle sammenhænge mellem de valgte klasser og hændelser.
Når objekterne, klasserne og hændelserne er identificeret, udarbejdes der en hændelsestabel og et klassediagram. Til sidst udarbejdes der nogle tilstandsdiagrammer over de forskellige tilstande, som kan forekomme i klasserne. Der henvises til del to i bogen Objekt Orienteret Analyse & Design [SAD1].

2.4.1 Hændelsestabel

Hændelsestabellen bruges til at skabe en bedre forståelse af, hvordan diverse klasser interagerer i forhold til diverse hændelser.
I nedenstående skema er der defineret klasser og hændelser, som er relevante i forhold til systemdefinitionen, samt operationen af systemet.

	Hændelser							
Klasser	Tilmelding	Afmelding	Resultat	Resultat-	Bane	Bane	Tilmelding	Afmelding
Resultatoversigt+				*				
Skytter +	+	+		*	+	+	+	+
Administrator +	*	*	*	*	*	*	*	*
Gruppe +				*				
Bane +					+	+		
Resultater +			+	*				
Disciplin +			+	*	+	+	*	*
Turnering +			*	*	+	+	+	+
Træning +			+	*	+	+		
Tid +					+	+		
Junior gruppe +				*				
Senior gruppe +				*				
Åben gruppe +				*				
Ung gruppe +				*				
Børne gruppe +				*				
Veteran gruppe +				*				
15 m +			+	*	+	+	*	*
Luft +			+	*	+	+	*	*
Klasse				*				
Skytteklub	+	+						

Tabel 1 - Hændelsestabel

Et eksempel på en klasse og dennes hændelser er klassen" Skytte" og de hændelser, denne klasse har. I hændelsestabellen ses det, at "Skytte" har en interaktion med hændelsen tilmelding og afmelding af klub. Disse hændelser er sat som værende engangshændelser, da den enkelte skytte kun kan afmeldes og indmeldes én gang. Indmeldes han igen på et senere tidspunkt, bliver han ansat for at være en ny skytte. Herudover ses det, at "Skytte" har noget med hændelsen resultatvisning at gøre, hvor det beskrives at denne interaktion vil forekomme flere gange.

Grunden er, at hver enkelt skyttes resultat vil blive vist mere end blot én gang. Afslutningsvis er der hændelserne bane booket og aflyst, samt tilmelding og afmelding til turnering. Disse interaktioner sker kun enkelte gange, da der aldrig er tale om den samme turnering eller den samme bane (banen har variabler som blandt andet tid, dato og bane nummer, der gør at der aldrig er tale om den samme bane med samme opsætning).

2.4.2 Klassediagram

For at strukturere klasserne og relationerne mellem dem, er der lavet et klassediagram, som ses på figur2. Klassediagrammet viser de forskellige klasser og hvordan de relateres til hinanden. I klassen ”Turnering” ses det, at en turnering er bestående af en eller flere reservationer, discipliner og grupper og denne kan derfor opstilles som en aggregeringsstruktur.

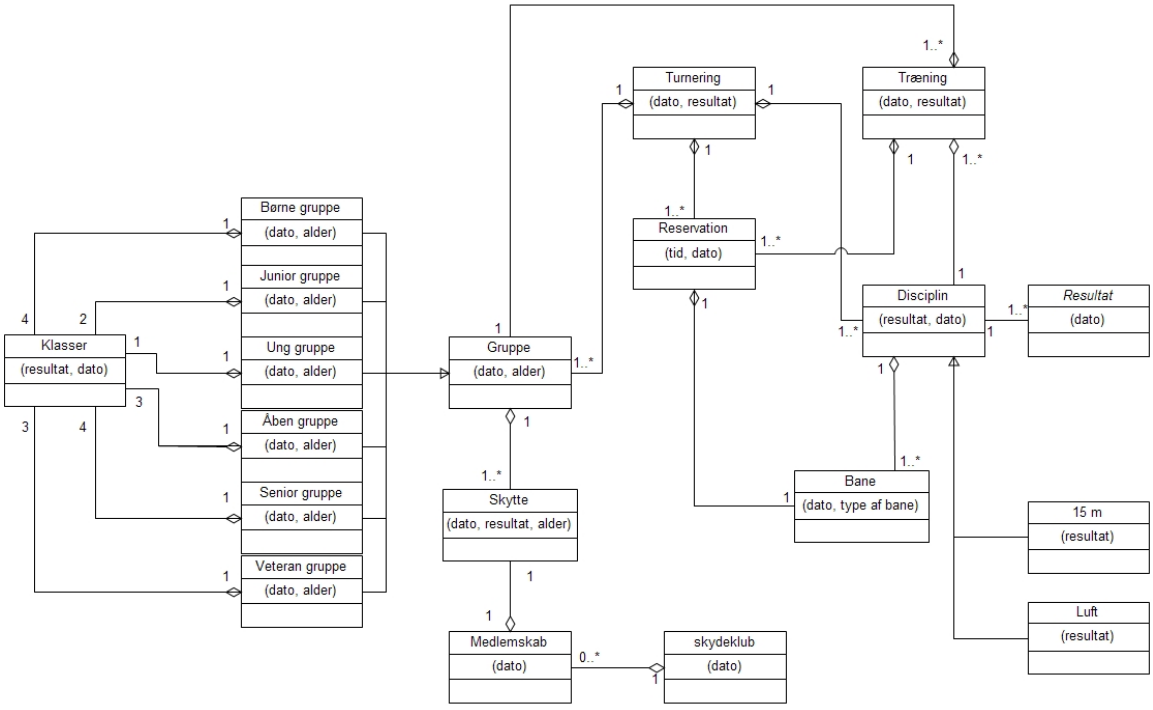
For at en turnering kan finde sted, skal der være styr på en række kriterier, eksempelvist hvilken skytte, hvilken bane der skydes på samt disciplinen der skydes i. Der skildres i denne klasse, at en bane er en del af reservation, da det er banen der reserveres.

Træningsklassen bliver tildelt denne samme struktur, da de er tilnærmelsesvis ens.

En disciplin kan enten være luft eller 15m, og følger dermed en generaliseringsstruktur, hvor disciplin er superklassen og luft samt 15m er subklasser. En disciplin består endvidere af en eller flere baner, da en disciplin ikke ville kunne finde sted uden en bane, så en bane er således en del af en disciplin. Til hver disciplin kan der forekomme én eller flere resultater.

Klassen ”Gruppe” er en superklasse bestående af en række subklasser: ”Børnegruppe”, ”Juniorgruppe”, ”Ungdomsgruppe”, ”Åbengruppe”, ”Seniorgruppe” og ”Veterangruppe”. Alle disse grupper indeholder et antal klasser alt afhængig af hvilken gruppe, der er tale om. Eksempelvist kan en børnegruppe indeholde fire klasser, en juniorgruppe to klasser og en ungdomsgruppe én klasse osv. [SAD2]

En skytte er en del af alle grupper, idet en gruppe består af en eller flere skytter, og disse skytter er så en del et medlemskab, som er associeret til skydeklubben.

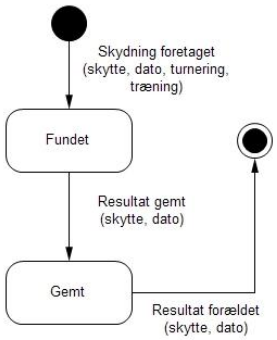


Figur 2 - Klassediagram

2.4.3 Tilstandsdiagrammer

Tilstandsdiagrammerne bliver brugt i analysen i problemområdet for, at give et visuelt overblik over de forskellige tilstande, som kan forekomme i de forskellige klasser.
Der ses også på hvilke ting der initialiserer hændelsen, hvilke hændelser der sker undervejs før terminering.
Nedenunder er der forklaringer til fire af diagrammerne. Disse 4 er udvalgt, da de giver et generelt overblik. Resten kan ses i bilag 1.

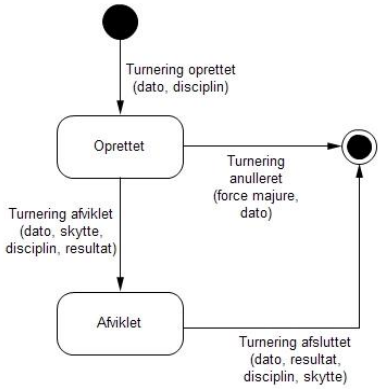
Resultat



Figur 3 - Tilstandsdiagram: Resultat

Resultatklassen initialiseres ved, at der er en skytte, som har foretaget en skydning og derved opnår et resultat. Resultatet er derfor ”fundet”. Det gemmes med en dato og skyttens skyttenummer. Det lagres i systemet og termineres først, når resultatet anses for at være forældet.

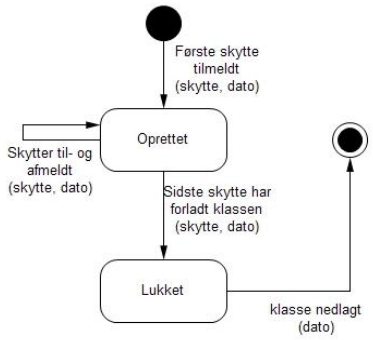
Turnering



Figur 4 - Tilstandsdiagram: Turnering

Denne initialiseres ved, at der bliver oprettet en turnering. Derved når den tilstanden ”oprettet”. Herefter afvikles turneringen og termineres når den er tilendebragt ellers bliver den annulleret, eventuelt grundet force majeure.

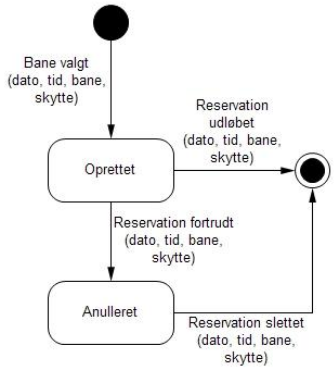
Klasse



Figur 5 - Tilstandsdiagram: Klasse

(Her menes der klasseinddeling for skytter og ikke for systemet)
Denne initialiseres ved at mindst én skytte tilmelder sig for at få oprettet klassen. Dermed opnår den tilstanden ”oprettet”. I denne tilstand opnår den forskellige til- og afmeldinger. Den terminerer først, når den sidste skytte har forladt klassen.

Reservation



Figur 6 - Tilstandsdiagram: Reservation

Denne initialiseres ved, at der vælges en bane og ledigt tidsrum. Når denne er færdiggjort får den tilstanden ”oprettet”. Herefter kan skydning så gennemføres og reservationen udløber derefter, når den reserverede tid udløber. Den anden form for termination er såfremt reservationen bliver annulleret, hvorefter tilstanden også bliver ”annulleret”. Dette frigiver banen igen, og ny reservation kan forekomme.

2.5 Analyse af Anvendelsesområde

Anvendelsesområdet er den del, der administrerer problemområdet. Aktørerne, der anvender systemet, identificeres og samtidig findes alle relevante arbejdsopgaver for hver enkel aktør.. Dette er særdeles vigtigt for, at få en forståelse for aktørerens behov i forskellige situationer de bruger programmet. Dette er vigtigt for at designe et effektivt og tilfredsstillende program til kunden. Der henvises til del tre i bogen Objekt Orienteret Analyse & Design [SAD1].

2.5.1 Aktørtabelen

For at fastlægge sammenhængen mellem aktører og deres brug, udarbejdes der først en aktørtabel for systemet.

	Aktører			
Brugsmønstre	Administrator	Skytte	Tilskuer	Receptionist
Tilmelding (turnering)		X		X
Afmelding (turnering)		X		X
Skytteinformation				X
Registrering af medlemskab				X
Reservering af bane		X		X
Resultatafvikling	X			
Resultatvisning	X	X	X	
Tilmelding (træning)		X		X
Afmelding (træning)		X		X
Annullering af bane		X		X

Tabel 2 - Aktørtabel

2.5.2 Aktørspecifikation

For at beskrive de enkelte aktørers rolle i forhold til systemet, udarbejdes en aktørspecifikation. En person kan godt have flere roller i forhold til systemet.

Der opereres med fire aktører: Administrator, Skytte, Tilskuer og Receptionist

Tilskuer
Formål: Disse personer skal kunne aflæse skytternes resultater på opsatte skærme til formålet.
Karakteristik: Tilskuerne kan i sig selv være meget forskellige, men i forhold til systemet bliver de anset for at være ens, da formålet for dem er det samme; nemlig at se en oversigt over skytternes resultater.
Eksempler: En tilskuer (A) har en bekendt med i en skydeturnering. Hun vil gerne se hvordan det er gået indtil videre i turneringen og kigger op på resultatskærmen. Her ser hun, at skyttens navn fremgår på skærmen og har klaret sig godt indtil videre, i forhold til de andre i hans klasse. En tilskuer (B) kigger på resultatskærmen. Han leder efter sin søns resultater. Skærmen viser dog ikke resultaterne for sønnens gruppe, men viser dagens resultater for veterangruppen. Tilskueren går fra skærmen med uforrettet sag, da han ikke vil vente mere.

Tabel 3 – Aktørspecifikation: Tilskuer

Skytte
Formål: Skytten er den indirekte bruger af systemet, idet det er ham som systemet er konstrueret til. Skytten bruger systemet til reservering af baner, til- og afmelding af træning, samt turneringer og for at se sine resultater.
Karakteristik Skytten bruger ikke systemet direkte, men benytter enten administratoren eller receptionisten til ovenstående formål. Skytten behøver dermed ikke større indsigt i IT, da han ikke interagerer med systemet direkte.
Eksempler Skytte (A) kontakter receptionisten i klubben i god tid og vil gerne booke en bane til træning på torsdag, i næste uge. Receptionisten tjekker i systemet og finder en bane til skytten. Skytten og receptionisten aftaler en tid og reservationen afsluttes. Samtidig minder receptionisten skytte A om en forekommende turnering 14 dage efter, hvilket skytten havde glemt og bliver tilmeldt med det samme. Efter endt træning, afleverer skytten selv sine resultater til administratoren, for herefter at se sine resultater på skærmen. Skytte (B) ringer onsdag til receptionisten og spørger, om der er en ledig bane til træning omkring kl. 17.00 dagen efter. Receptionisten tjekker i systemet og ser at der er booket. Receptionisten tjekker dagsoversigten i systemet og ser, at der først er en ledig bane fra kl. 18.30.

Tabel 4 – Aktørspecifikation: Skytte

Administrator
Formål: Denne persons formål er hovedsagligt, at sikre korrekt resultatafvikling, pointvisning samt fejlhåndtering.
Karakteristik: Systemet kræver, at der er en administrator, der får en introduktion til systemet. Han skal desuden have styr på reglerne indenfor skydning og være fortrolig med systemets brugergrænseflade, for at sikre korrekt afvikling af resultater.
Eksempler: En erfaren administrator (A), der kender til almen IT og er bekendt med lignende systemer. Såfremt der kommer en fejl i systemet går han ikke i panik, men foretager en systematisk fejlfinding. Han er erfaren indenfor skydesporten, så han ved hvilke resultater der skal indtastes og hvor. Han har overblik over systemets begrænsninger og muligheder. En uerfaren administrator(B) er kun lettere erfaren indenfor IT. Det tager lang tid før administratoren føler sig fortrolig med systemets brugergrænseflade, da han ikke kan forstå systemets opbygning. Ved fejl i systemet prøver han at genstarte og prøve igen, og kontakter derefter producenten. Han er bekendt med skydesportens regler, men mangler fuldt overblik over hvor det skal indtastes i systemet.

Tabel 5 – Aktørspecifikation: Administrator

Receptionisten
Formål: Receptionisten har til formål at tilmelde og afmelde skytter til henholdsvis turnering og træning, hvorpå skydebanerne reserveres. Endvidere sørger receptionisten for registrering af skytterne med tilhørende skytteinformation.
Karakteristik: Receptionistens har en bred viden om IT, hvorpå vedkommende hurtigt og enkelt skal kunne anvende systemet til sine arbejdsopgaver.
Eksempler: Receptionist (A) har et bredt overblik over sine arbejdsopgaver og kan hurtigt løse sine opgaver ved anvendelse af systemet. Receptionist (A) er ligeledes bekendt med diverse IT-systemer, hvorpå de forskellige data indtastes korrekt. Receptionist (B) er ikke så IT-erfaren, hvilket medføre at nogle arbejdsopgaver tager længere tid at løse. Dette bevirker, at receptionist (B) ofte prøver sig forsigtigt frem, hvorefter hun spørger om hjælp, hvis opgaven endnu ikke er løst. Receptionisten vil ligeledes være nervøs for om indtastningerne er korrekte, i forhold til det forventede.

Tabel 6 – Aktørspecifikation: Receptionisten

2.5.3 Brugsmønstre specifikation

Nu da aktørspecifikationen er blevet beskrevet, kan der derefter blive beskrevet de forskellige brugsmønstre, som har til formål at vise, hvordan der bliver interageret med systemet.

Turnerings tilmelding
Mønster: Turneringstilmelding forekommer når en skytte ønsker at deltage i en kommende turnering. Skytten henvender sig til receptionisten, som indtaster relevante skytteinformationer og derefter reserverer en plads i turneringen.

Turnering afmelding
Mønster: Turneringsafmelding opstår i det øjeblik en skytte er tilmeldt en given turnering, men ønsker at afmelde sig denne. Skytten henvender sig til receptionisten for at tilkende give dette, receptionisten afmelder den pågældende skytte og sikre sig, at der ikke reserveres en plads til denne skytte.

Skytteinformation
Mønster: Når der bliver oprettet en ny skytte i klubben, indtaster receptionisten relevante informationer i systemet om den pågældende skytte, og det er kun receptionisten, der har adgang til disse informationer.

Registrering af medlemskab
Mønster: Når en skytte bliver oprettet, skal dette nye medlemskab registreres i systemet, dette er receptionistens opgave.

Reservering af bane
Mønster: En skytte vil gerne reservere en bane og receptionisten foretager reservationen af den valgte bane til den pågældende skytte, såfremt den er ledig.

Annullering af bane
Mønster: En skytte vil gerne afmelde en bane og receptionisten finder skyttens reserverede bane frem og sørger for, at annullere reserveringen.

Resultatafvikling
Mønster: Resultatet indsamles efter hver skytte har gennemgået en skydesektion, derefter indtastes resultatet af administrator. Resultatet bliver bearbejdet i systemet og tillægges eventuelle tidligere relevante skyderesultater for den pågældende skytte.

Resultatvisning:
Mønster: Pointvisningen bliver udarbejdet af administrator. Efter resultatafvikling bliver den samlede score opdateret for hver enkel skytte og sammenlignet med andre skytter, i samme træningsklasse eller turnering. Pointvisningen bliver offentliggjort for tilskuere og skytter.

Trænings tilmelding
Mønster: Trænings tilmelding forekommer, nå en skytter ønsker at deltage i en opkommende træningsaften. Skytten henvender sig til receptionisten, som indtaster relevante skytteinformationer og derefter reserverer en bane.

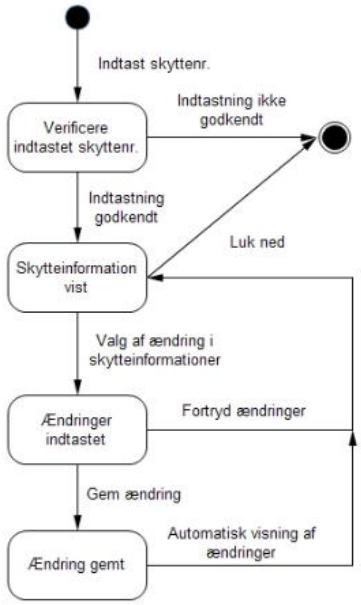
Trænings afmelding
Mønster: Trænings afmelding opstår i det øjeblik en skytte er tilmeldt en given træning, men ønsker at afmelde sig denne. Skytten henvender sig til receptionisten for at tilkendegive dette. Receptionisten afmelder den pågældende skytte og sikre sig, at der ikke er reserveret en bane til denne skytte.

2.5.4 Tilstandsdiagrammer for brugsmønstre

Tilstandsdiagrammer for de forskellige brugsmønstre er medvirkende til forståelsen for, hvordan brugeren interagerer med systemet.

Skytteinformation

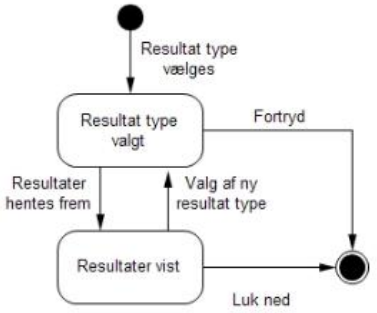
I det første brugsmønster er der fokus på receptionisten, som skal behandle skytternes informationer. Først indtastes et skyttenummer, hvorefter systemet tjekker om indtastningen er korrekt. Hvis dette ikke er tilfældet, vil funktionen terminere og receptionisten kan på ny søge efter skytten. Hvis indtastningen derimod er korrekt, kan receptionisten se skyttens informationer, hvorefter der kan vælges at lukke programmet eller ændre i de data, som systemet indeholder. Herunder er det også muligt at afmelde skytter fra klubben. Under ændringen kan receptionisten både fortryde eller gemme sine ændringer, hvorefter der føres tilbage til skytteinformationerne.



Figur 7- Tilstandsdiagram for Skytteinformation

Resultatvisning

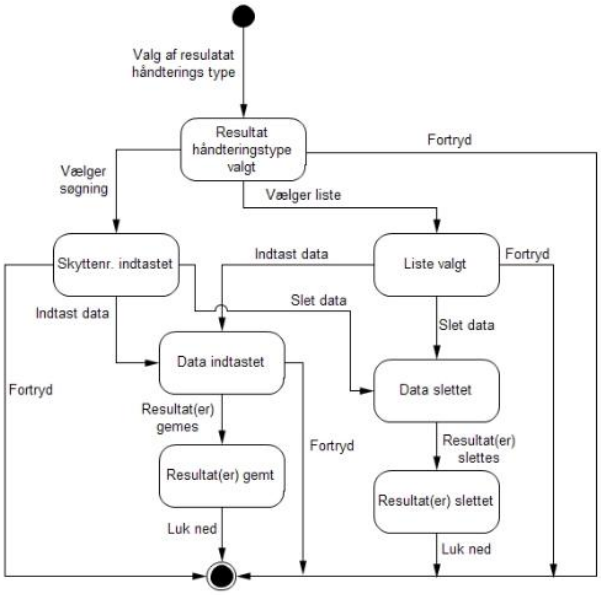
Ved resultatvisning skal der vælges hvilken type der ønskes vist, hvorefter det vises på skærmen. Ligeledes kan der bevæges mellem valg og visning, hvis der ønskes at hente andre resultater frem. Denne funktion kan ligeledes termineres i alle situationer.



Figur 8 - Tilstandsdiagram for Resulatatvisning

Resultatafvikling

Under resultatafviklingen skal der først vælges, hvilken håndteringstype der ønskes at arbejdet med, hvorefter der vælges om der ønskes at søge eller se på lister. Herefter kan der vælges at indtaste eller slette resultater og gemme ændringer. I alle situationer kan valget fortrydes, hvis der indtastes noget forkert.

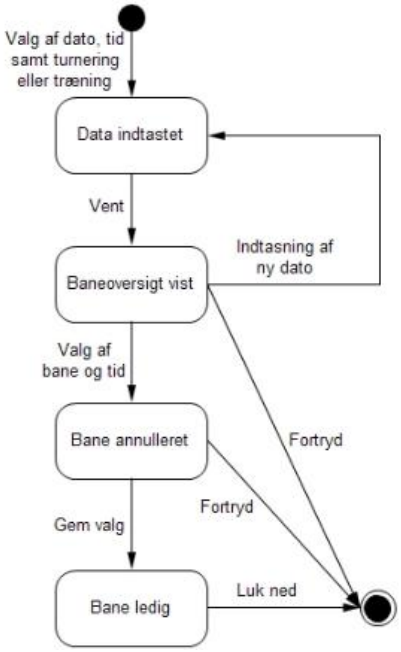


Figur 9 - Tilstandsdiagram for Resultatafvikling

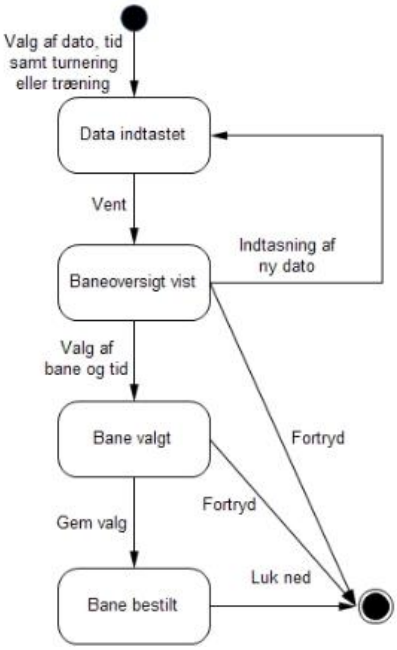
Bane og aktivitets tilmelding og annullering

Nedenstående to diagrammer er ens opbygget. I "reservering af baner"-diagrammet indgår reservering af baner, hvorpå tilmelding til turneringer og træning indgår. Først indtastes en dato, hvorefter systemet viser om banen er ledig på det givne tidspunkt. Er dette ikke tilfældet, så kan en ny dato indtastes. Hvis banen er tilgængelig, så kan der reserveres og valget kan gemmes. Det er samtidigt muligt at fortryde sine valg, hvorefter systemet terminerer.

Annullering og afmelding af baner til træning og turnering foregår på samme måde som reservationen, da de samme trin skal gennemgås for at annullere.



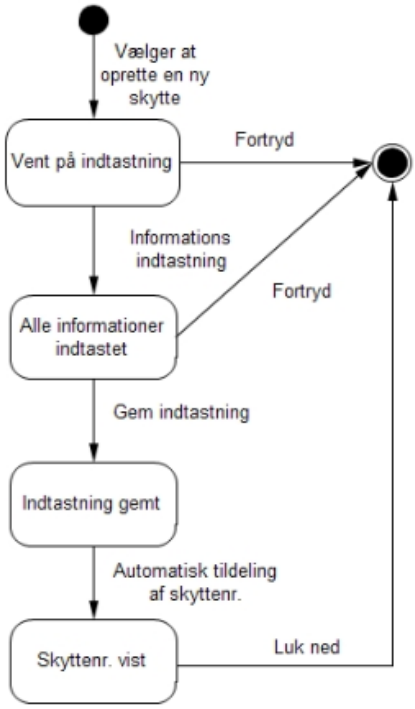
Figur 10 - Tilstandsdiagram for bane og aktivitet annulleret



Figur 11 - Tilstandsdiagram for bane og aktivitet tilmeldt

Registrering af ny skytte

Ved registrering af en ny skytte begynder, skal der først foretages valg om, der ønskes at oprette en ny skytte i systemet. Derefter indtastes de relevante informationer, hvorefter de gemmes. Der tildeles automatisk et skyttenummer, som vises på skærmen når oprettelsen er fuldendt. På den vis kan receptionisten hurtigt give den nye skytte hans eller hendes nummer. Undervejs kan der vælges at fortryde både før og efter indtastningen af informationer.



Figur 12 - Tilstandsdiagram for Registrering af ny skytte

2.5.5 Funktionsliste

En funktionsliste viser de fundne funktioner i systemet, samt deres kompleksitet og type.

Funktionerne er inddelt i tre funktionstyper: Opdatering, aflæsning og beregning. Alle funktionerne er også inddelt i kompleksitet. I dette system er der to funktioner, som er af medium sværhedsgrad, nemlig reservering af bane og resultatafvikling. Disse to funktioner er af henholdsvis opdaterings- og beregningstypen. Disse to funktioner er de mest vigtige, så derfor er det vigtigt, at systemet skal kunne reservere, opdatere baner, beregne resultater og holde styr på disse.

Resten af funktionerne er af simpel sværhedsgrad, da disse funktioner ikke er særlig komplekse i systemet.

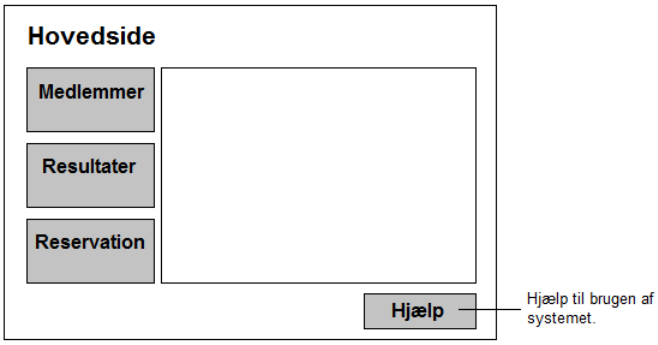
Funktion	Kompleksitet	Funktionstype
Registrering af medlemskab	Simpel	Opdatering
Tilmelding (turnering)	Simpel	Opdatering
Afmelding (turnering)	Simpel	Opdatering
Reservering af bane	Medium	Opdatering
Annullering af bane	Simpel	Opdatering
Resultatvisning	Simpel	Aflæsning
Resultatafvikling	Medium	Beregning
Skytteinformation	Simpel	Aflæsning
Tilmelding (træning)	Simpel	Opdatering
Afmelding (træning)	Simpel	Opdatering

Tabel 7 - Funktionsliste

2.5.6 Vinduesdiagrammer

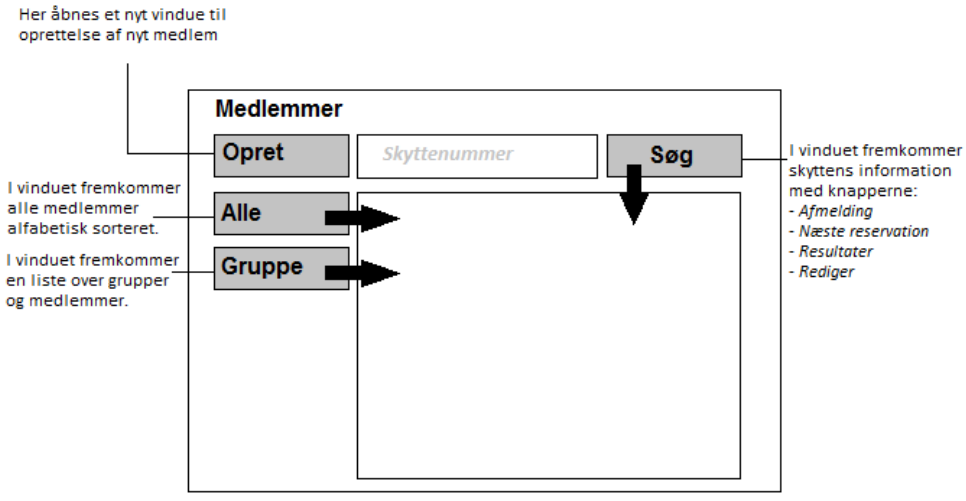
Vinduesdiagrammer er skitser til systemets brugerflade og har til formål at give en idé om, hvad de enkelte vinduer skal indeholde.

I vinduesdiagrammet til højre er hovedsiden anvist, dette er menuen, aktøren først bliver introduceret for. Hoved-siden er omdrejningspunktet for programmet. Her er det muligt for aktøren, at navigere rundt i systemet, på en nem og overskuelig måde. Uanset arbejdsopgave er dette siden, der henviser til det relevante sted i programmet for en given opgave.



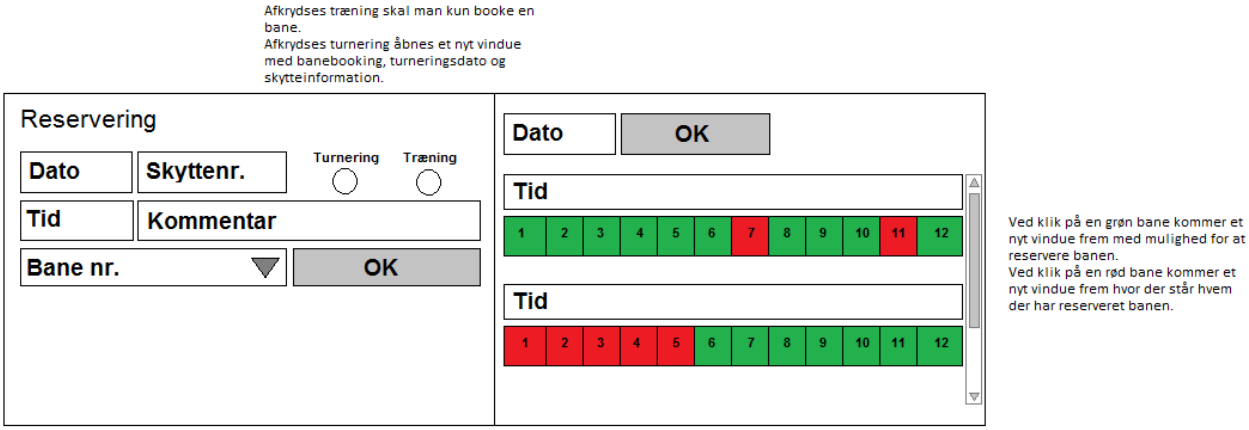
Figur 2 - Vinduediagram: Hovedside

Her ses vinduesdiagrammet for medlemmer, her er det muligt, at søge på et skytte-nummer. Dette kan resultere i at en given skyttes informationer bliver anvist i boksen, som vist på diagrammet. Ligeledes er det her i systemet muligt at redigere eller slette i skytteinformationer, samt oprettelse af nyt medlem.
Vinduet medlemmer giver også mulighed for at udskrive en liste over alle medlemmer eller grupper.



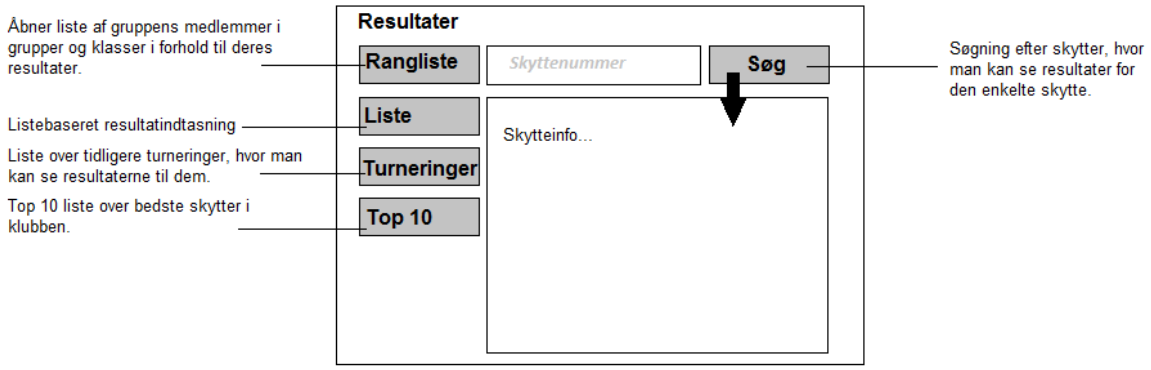
Figur 14 - Vinduediagram: Medlemmer

I vinduesdiagrammet set nedenfor er vinduet for reservationen anvist. Her er det mulig at lave en bane tilmelding, til en træning eller turnering på en given dato. Systemet tilbyder også at søge en dato, dette giver muligheden for at få en liste over ledige eller reserveret baner fremvist.



Figur 15 - Vinduediagram: Reservationen

Sidste er resultat-vinduet vist, her kan der søges efter skytteresultater. En søgning kan foregå manuelt via skyttenummer eller gennem en rangliste, liste, turnering eller top 10 visning. Ligeledes er det muligt, at redigere i resultaterne.

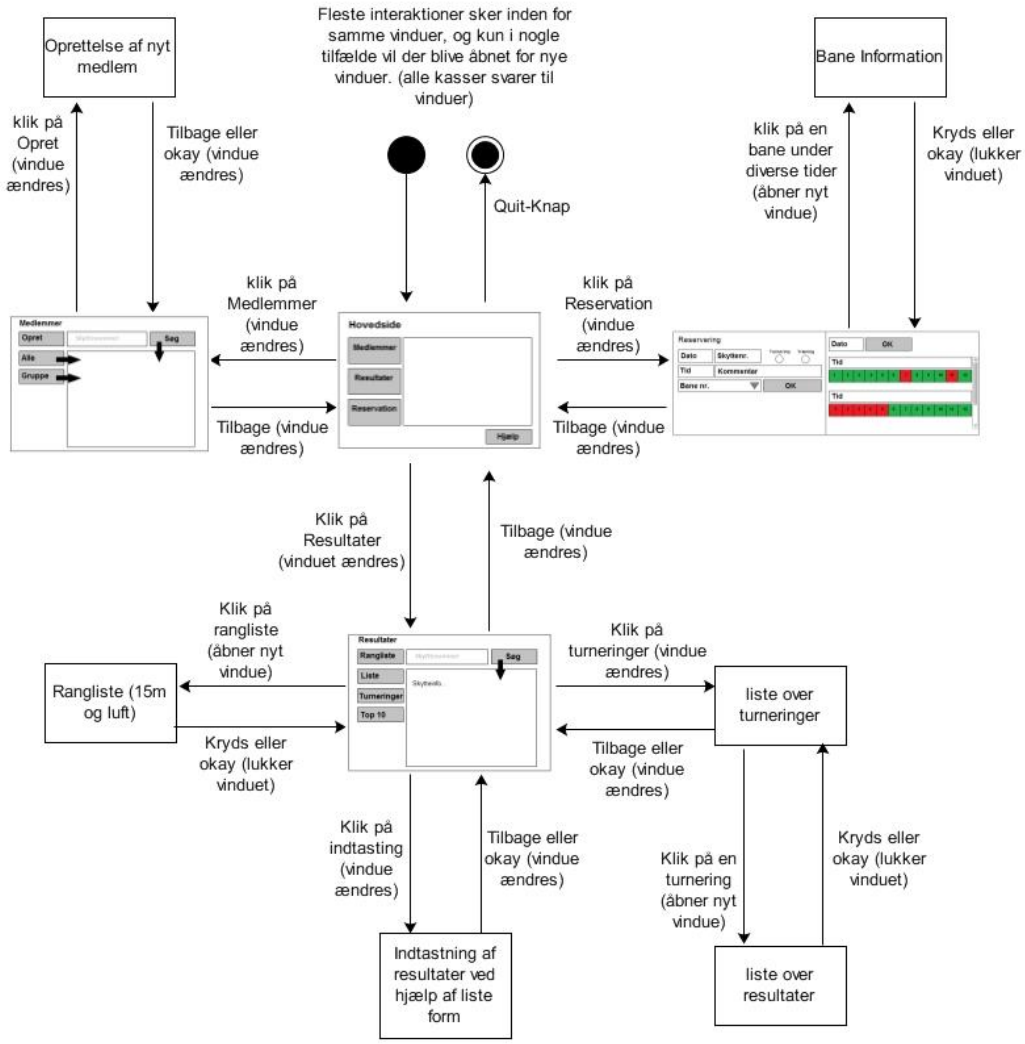


Figur 16 - Vinduediagram: Resultater

2.5.7 Navigationsdiagram

Navigationsdiagrammet viser de dybtgående vinduer og overgangene mellem dem.

Ved at skildre dynamikken i brugergrænsefladen som en helhed, kommer der en dybde og bredde ind i systemudviklingen. Et vindue har til formål at beskrive en tilstand i systemet, med navn og indhold. Indholdet består i form af en miniature af de enkelte vinduer i systemet, samt en menu overskrift. Nedenstående diagram skildrer et vindueskift og de dertilhørende tilstandsskift. Diagrammet beskriver, hvordan tilstandene knytter sig til vinduerne og hvordan en aktør kan navigere mellem de forskellige menuer og vinduer i systemet, ved at klikke på et ikon.



Figur 17 - Navigationsdiagram

Navigationsdiagrammet viser, at der er valgt en brugergrænseflade med et hovedvindue hvori der er tre ikoner med undermenuer og ét ikon med hjælp. Et tryk på hver af de tre undermenuer resulterer i et vindue og tilstandsskifte.
Ikonet *Medlemmer* medfører vinduesskifte med overskriften *Medlemmer*. I dette vindue kan der tilmeldes en ny skytte, søges en enkel skytte, se alle skytter eller se grupper af skytter.
Det andet ikon *Reservation* fører til vinduet med overskriften *Reservering*. Her kan der reserveres eller afmeldes en reserveration, og der er en oversigt over ledige og optagede baner i et givent tidsinterval.
Tredje ikon *Resultat,* er det mest omfattende. Her er mulighed for at indtaste nye resultater, hvad enten det er træning eller turneringsresultater. Ydermere kan aktørerne se ranglisten over bedste skytter i disciplinerne 15m og luft skydning. Der kan også vises lister over turneringer og dertilhørende turneringsresultater. I dette vindue er der også et søgefelt, hvor der kan søges efter en skytte via et skyttenummer. Herfra kan den givne skyttes resultater så fremvises.

2.6 Design af arkitektur

Formålet er her at strukturere systemet. Ved denne fokuseres der blandt andet på struktureringen af systemets komponenter og processer. Der henvises til del fire i bogen Objekt Orienteret Analyse & Design [SAD1].

Kriterium	**Meget vigtigt**	**Vigtigt**	**Mindre vigtigt**	**Irrelevant**	**Trivielt opfyldt**
Brugbart	X				
Sikkert					X
Effektivt		X			
Korrekt		X			
Pålideligt					X
Vedligeholdbart			X		
Testbart				X	
Fleksibelt			X		
Forståeligt	X				
Flytbart					X
Integrerbart		X			

Tabel 8 – Dsign af arkitektur

2.6.1 Prioriteringsskema

I prioriteringsskemaet er der opstillet nogle kriterier, hvor der vægtes på, hvor vigtige de er i forhold til systemet. Nedenunder gives der eksempler på nogle enkelte af kriterierne, og forklaret hvorfor de er vægtet som de er.

Forståeligt er registreret som "Meget vigtigt", da dette er meget vigtigt i forhold til dem der skal bruge systemet. Aktørerne af systemet er blandt andre ældre brugere, og disse brugere kan have en tendens til, at have svært ved at forstå it-systemer generelt.

For at tage et andet eksempel fra skemaet, kan der kigges på kriteriet "Flytbart". Her vurderes dette kriterium til at være "Trivielt opfyldt". Grunden til dette er, at dette er ligegyldigt for systemet, da systemet blot skal stå stationært nede i et af skydeklubbens lokaler.

Spørgsmål	Model	Funktion	Grænseflade
Ansvar	Model af problemområde	Funktionalitet på modellen	Interaktion mellem funktionalitet og brugere eller andre systemer
Omgivelser	Simpelt problemområde med enkle opdelinger	Behov for simple funktionsudregninger til registrering.	Enkel resultatvisning til aktørerne; Enkel indtastning for administrator
Eksempler	Reservation, inddeling, træning	Beregning, rangering, reservering	Vindueinddelt grænseflade
Specielle krav	Organisering	Hurtig beregning, sikre resultater.	Skærme, vinduer, knapper,

Tabel 9 - Prioriteringsskema

2.6.2 Komponent-Skema

Skemaet viser en simpel oversigt over de tre komponenter og deres funktion og ansvarsområde i systemet. Hver komponent har sit ansvarsområde i systemet, og derfor hjælper skemaet til bedre forståelse af de enkelte komponenter, og hvordan de skal bruges i systemet.

Nedenstående model er for de enkelte komponenter og viser hvordan de er implementeret i systemet, og hvordan de interagerer med hinanden.

De tre komponenter der bliver anvendt er:

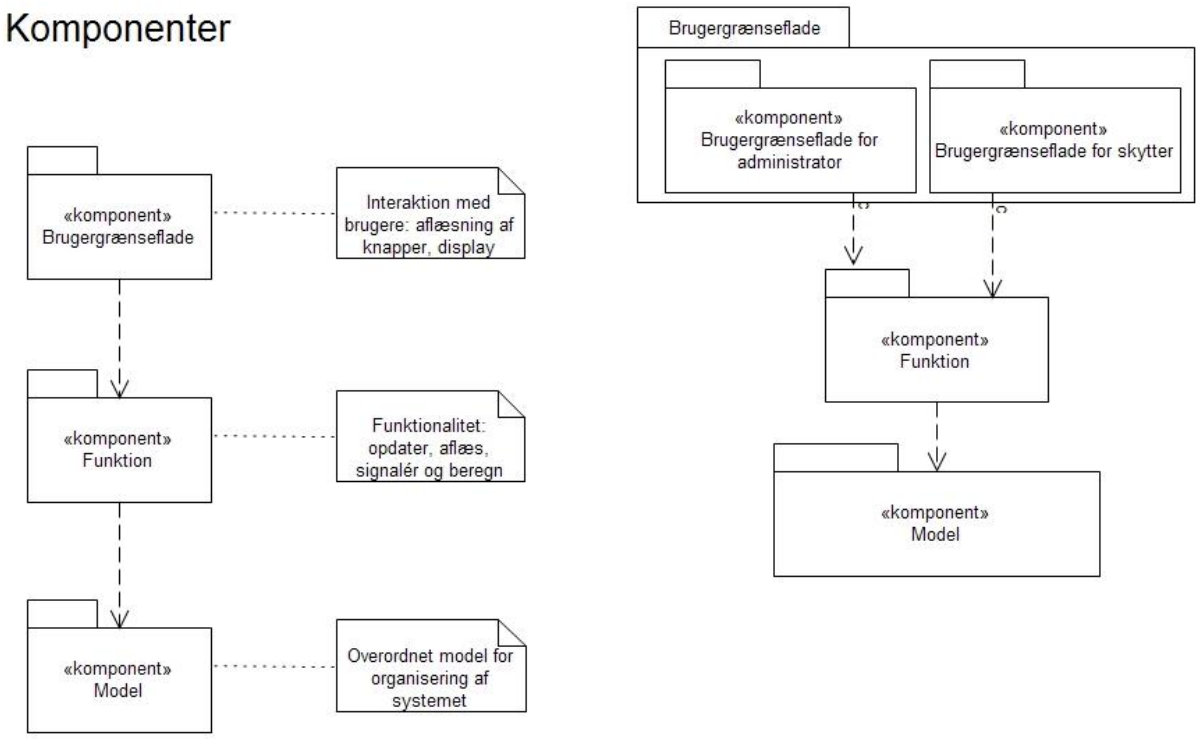
1. Grænsefladen
2. Funktion
3. Model

Grænsefladen er det, som aktørerne ser, samt det administrator/receptionisten interagerer med. Da der er to grænseflader, er denne komponent delt op i 2 ”under-komponenter”.

Funktionskomponenten er de beregninger, der foretages i systemet.

Model-komponenten er oversigten over systemet.

De to grænseflader er begge en del af funktions-komponenten, hvilket er afbilledet i modellen. Funktions-komponenten er en del af modelkomponenten, som også er afbilledet.



Figur 18 – Komponent-skema

2.6.3 Delkonklusion

Efter endt gennemgang af analyse og design-afsnittet, er der opstillet en række skemaer og diagrammer. Disse hjælper til at give overblik og bedre forståelse af systemets opbygning. Illustrationerne er resultatet af analyse af de 3 hovedområder: Problemområde, anvendelsesområde samt design af arkitektur.

I problemområdet er der fastlagt de basale krav til systemet og inddelt systemet i klasse. Dette krævede nøje gennemgang af funktionerne, som skytteklubben har brug for, samt de tilhørende regler. Ud fra klasserne er de hændelser, der kan ske, blevet fundet og tilstandsdiagrammer udarbejdet, for at give en oversigt over dette.

I anvendelsesområdet er der analyseret mere på hvordan aktørerne interagerer med systemet, hvem de er og deres brugsmønstre. Dette gav et godt overblik i forhold til aktørernes synspunkter og sikrer derved at de kan bruge systemet. Således er der også skitseret, hvilke funktioner som vinduerne skal indeholde. Dette har medført, at der kunne tegnes et navigationsdiagram, så man ved hvordan man navigerer rundt i systemet.

I design og arkitekturafsnittet er der udarbejdet en prioriteringsliste og komponentliste- samt skema, hvilket giver en oversigt over, hvilke ting, der skal fokuseres på under udviklingen, samt hvilke komponenter der er med.

3 Brugergrænseflade

Dette kapitel beskriver arbejdet med design af brugergrænseflade, hvilket udformes i Windows Forms. Brugergrænsefladen vil blive implementeret i en prototype, som er et uddrag af systemet. I forbindelse med designet af brugergrænsefladen, er der blevet undervist i kurset "design, implementering og evaluering af brugergrænseflade" (DIEB). Målet med DIEB kurset var, at bruge det der blev lært under forelæsningerne, til at designe brugergrænsefladen for en prototype af systemet.
Målet for programmets brugergrænseflade er, at aktørerne finder det nemt, effektivt og tilfredsstillende at bruge programmet.
Under udviklingen af brugergrænsefladen skal slutbrugernes evner, arbejdsopgaver og behov inkorporeres i udviklingen. Såfremt det er personer med relativ lav IT-erfaring, er det nødvendigt at lave denne brugergrænseflade mere enkel og overskuelig.
En god brugergrænseflade er afgørende når et program skal opfylde de fastlagte krav, da dette ellers vil medføre frustration hos aktørerne. En god brugergrænseflade sikrer en intuitiv opfattelse af systemet hos aktøren, og bevirker at personen kan interagere med systemet på en logisk måde. Da systemet ikke er et "ekspert-system", er det vigtigt at brugergrænsefladen er enkel og overskuelig.
Da brugergrænsefladen er udarbejdet i Windows Forms, vil dette hjælpe med genkendelse og hurtigere fortrolighed hos aktøren, da Windows er et af de mest brugte styrersystemer. [DIEB1]
Derudover er der i udarbejdelsen af brugergrænsefladen lagt vægt på konsistens og mapping. Dette gøres for, at sikre at aktøren bedre kan forholde sig til, hvad de enkelte funktioner og knapper gør.
Der er planlagt en lille usabilitytest med en aktør fra selve skydeklubben. Formålet er at se, om designet passer til aktørernes behov og for at se, om de finder det logisk at bruge.

3.1 Metode

I denne case vægtes en videre implementering af SAD modulets design ideer. Interaktionsformer anvendes til at evaluere en opbygning af designet, og systemets interaktion med aktøren. Mockups vil blive anvendt under evalueringen, disse vil udgøre det endelige design. Systemets interaktion vægtes bl.a. gennem en stakeholder analyse og personas model. En gennemgående analyse af aktørerne vil bidrage til en identifikation af alle personer, som eventuelt vil interagere med systemet. Ydermere kan der analyseres på hvilke behov den enkelte aktør har i forhold til systemet.

Med kendskab til aktørerne, kan en prioriteringsliste danne et billede af funktionskravene til systemets brugergrænseflade. Gestaltlovene og de fire designprincipper vil blive anvendt til at skabe en logik for aktørerne, når der navigeres gennem systems brugergrænseflade.
Det forventes at systemudviklingen vil undergå visse, mere eller mindre omfattende ændringer omkring design og interaktionen. I forbindelse med dette, vil systemets interaktionsrum ligge grund for visse ændringer i systemet, da denne metode ikke var implementeret i begyndelsen af designfasen.
Brugergrænsefladen vil afslutningsvis undergå en usabilitytest, testresultatet vil analyseres og evalueres på og indgå i diskussionen.

Usabilitytest

Testens roller vil bestå i en testleder og en datalogger. Testlederen har til formål, at sørge for at testen går som den skal. Han hjælper ikke direkte testpersonen, hvis han eller hun skulle gå i stå under selve testen. Testlederen sørger ligeledes for, at minde testpersonen om at tænke højt, således at dataloggeren kan notere alle tanker omkring anvendelsen af systemet. Dataloggeren har til formål at notere alle relevante observationer, som opstår under interaktionen. Disse noter uddybes således at dataudviklerne kan rette de problemer, der end måtte være ved systemet. Endvidere vil der efter testen blive foretaget en række Efter-sessionsspørgsmål, hvor testpersonen har mulighed for at give sin overordnede mening til kende, i forhold til prototypen.
Under analysen vil der blive benyttet en teknik, kaldet Instant Data Analysis (IDA). Denne metode foregår således, at der er en IDA-facilitator, som ikke var til stede under selve testen. Facilitatorens primære opgave består således i at opskrive problemerne, som fandt sted ved interaktionen ud fra de informationer, vedkommende får fra testlederen og dataloggeren. Dette skal bidrage til, at de mest alvorlige usability problemer i systemet bliver identificeret. Denne metode erstatter ligeledes videoanalyse, hvilket gør denne metode mindre ressourcekrævende. [DIEB2]
Måden hvorpå problemerne karakteriseres, ses i form af en tabel, som har kategorierne kosmetisk, alvorlig og kritisk. Disse problemer bliver bedømt via tiden det tager, at løse opgaven, irritationen ved benyttelsen og forskellene mellem forventning og programmets funktion. Da der kun er én testperson til usabilitytesten, kan de fundne problemer ikke være fyldestgørende. Skemaet der benyttes er angivet nedenfor.

	Delay	Irritation/ irrationality	Expectation vs. actual
Cosmetic	< 1 minute	Low	Small diff.
Serious	Several minutes	Medium	Significant diff.
Critical	Total (user stops)	Strong	Critical diff.

Figur 19 – taget fra UUIT_lecture3_4.pdf, udleveret af Jesper Kjeldskov, Vejleder

Da der ikke er mere end én testperson, vil der ikke blive bedømt ud fra den 4. kategori (katastrofe).

3.2 Afgrænsning

Der bliver i denne case afgrænset fra, at se på de dele af programmet, hvori medlemsadministrering og banereservation indgår, idet resultatindtastning og visning er blevet fundet mest interessant. Det menes derfor også, at der er mulighed for at arbejde med forskellige former for programmering, som kan give mulighed for udvikling og læring.
Ligeledes bliver der afgrænset fra at foretage en større usability test af prototypen med videoanalyse og en større vifte af testpersoner. Dette skyldes, at der er blevet undervist i metoden IDA, som skal bidrage til at denne del ikke tager så lang tid.

3.3 Prototypen

Det administrative system, skal kunne administrere medlemmer, foretage banebooking og håndtere resultater. Den valgte programmerede del af systemet, bliver en prototype, som kun håndterer én af de tre førnævnte hovedfunktioner, navnlig resultathåndteringsdelen. Dette er valgt ud fra flere argumenter. Resultatdelen er en meget vigtig del af administreringen på skytteklubben, men den del er ikke implementeret godt nok i deres nuværende system. Resultatfremvisning foregår nu i form af en tavle, hvorpå der er hængt et stort antal papirark med resultater. Derfor vil det være interessant at implementere det i systemet, således at resultatfremvisning kan foregå elektronisk i form af en slags diasshow. Det vil derfor også være interessant at arbejde med to forskellige brugergrænseflader, en indtastnings- og redigeringsgrænseflade for administratoren og en aflæsningsgrænseflade for skytter og tilskuer.

3.4 Begrebsmæssig model

Systemet har til formål, at hjælpe brugerne i en skytteforening med at administrere medlemmer og resultater. Det skal kunne vise skyttens bedste resultater og medvirke til flytninger i grupper og klasser, således at skytten kan se sine forbedringer via en ekstern fremvisning. Systemet skal endvidere gøre det let for administratoren eller receptionisten, at indtaste nye resultater til den enkelte skytte, enten via søgning eller lister. Med et enkelt design, skal det være muligt for aktøren at navigere rundt og interagere med systemet, hvilket skal medvirke til at det går hurtigt at indtaste resultater. Systemet vil kunne beregne de ti bedste resultater pr. skytte, så de får den korrekte placering på top ti og ranglisten.
Brugeren aktiverer de forskellige funktioner via knapper, som skal opfylde det som aktøren forventer af systemet. Dette er når aktøren aktiverer en funktion, som eksempelvis søgning efter skyttenummer. Systemet vil udføre søgefunktionen og derefter returnere et output. Aktøren bevæger sig ligeledes igennem en samling af informationer og udvælger derefter de dele af dem, der er relevante i forskellige arbejdssituationer. [DIEB9]

3.5 Brugeranalyse

Brugeranalyse handler om at finde de aktører, der har interesse i systemet og som eventuelt skal interagere med det.

3.5.1 Stakeholders

Systemets stakeholders er de personer, som har interesse i systemet, dvs. skytter, administratorer, selve klubbens medlemmer osv. Analyse af disse personer skal hjælpe til at forstå brugeren af systemet, således at et optimalt design kan blive fremstillet. De primære stakeholders er administratoren og receptionisten, som kan være den samme person eller flere forskellige. De sekundære stakeholders ses i form af skytterne eller tilskuerne, som ser resultaterne på en ekstern skærm. Der er ligeledes også tertiære stakeholders, som er andre skytteforeninger og deres administrerende system til resultatafvikling og medlemskab. Dette programs fokus bliver dog rettet mod de primære og sekundære stakeholders, da det er disse brugere systemet designes til. [DIEB3]

3.5.2 Personas

I forlængelse af stakeholderanalysen vil der nu blive analyseret nærmere på personas, som er en teknik, hvor man identificerer de typiske brugere mere detaljeret. I den forbindelse beskrives en fiktiv person, som ville være en typisk bruger af systemet. Beskrivelsen af denne er person er ikke valgt ud fra tilfældige fakta, men vurderet ud fra den gennemsnitlige medarbejders alder, job og computererfaring på skytteklubben. [DIEB3]

3.5.2.1 Karakteristik

Navn: Finn Jensen

Baggrund: Han er 53 år gammel, er gift og har 2 børn. Han har tidligere uddannet sig indenfor Lager og Logistik, og han arbejder nu som lagermedarbejder i en mellemstor virksomhed. I hans fritid går han meget i den lokale skydeklub, da han interesserer sig meget for skydning.

Computererfaring: Finn er begynder indenfor computerverdenen. For Finn skal det være nemt og ligetil. Den eneste erfaring han har med andre systemer er lagersystemer, samt lidt privat brug såsom e-mails og mindre surf på internettet. Han er meget gammeldags, og går derfor ikke meget op i computere, men han fik anskaffet sig én for et par år siden pga., at hans ene datter bor i London, og derfor holder han kontakten med hende via e-mail. Finns præferencer til det nye system er, at det skal være så simpelt som muligt, så han kan finde ud af at bruge det.

Den ovenstående persona er et eksempel på en typisk bruger af systemet. Aldersgrænsen går fra 18 år og opefter. Brugeren skal ikke være en avanceret computerbruger, for at kunne bruge systemet.
Der er således tale om en ældre arbejdende mand, som ikke har den store ekspertise indenfor brug af IT.

3.6 Mål for interaktion

Når der udvikles et IT-system, er det vigtigt for udviklingsholdet at sætte sig nogle mål for IT-systemet, der skal udvikles. Disse mål er blandt andet mål for interaktion, og herunder kigges der på mål for usability, samt mål for experience (oplevelse). Det er vigtigt, at udviklingsholdet prioriterer målene alt efter hvor vigtige de netop er for det IT-system, der skal udvikles. En prioritering er vigtig for ikke at skabe konflikter under udviklingen. Eksempelvist kan det give unødvendigt tidspres i forhold til hvornår IT-systemet skal være færdigt, som kan resultere i et færdigt produkt, der ikke tilfredsstiler slutbrugerne af produktet.
Herunder er der opstillet de mål og prioriteringer, i forhold til skytteklubbens administrationssystem.
Skemaet er skrevet efter det skema, der er undervist i, i DIEB kurset [DIEB4].

		Very Important	Important	Less important	Irrelevant
Usability	Effectiveness		X		
	Efficiency		X		
	Safety				X
	Utility		X		
	Learnability		X		
	Memorability	X			
Experience	Satisfying		X		
	Enjoyable			X	
	Fun				X
	Entertaining				X
	Helpful	X			
	Motivating			X	
	Aesthetically pleasing		X		
	Supportive of creativity				X
	Rewarding			X	
	Emotionally fulfilling				X

Tabel 10 – Skema for mål for interaktion

3.6.1 Usability

Målet ”memorability” er prioriteret som meget vigtigt, da medlemmerne i skytteklubben er aktørerne. Disse personer er typisk af den ældre generation, som ikke er så trygge ved brugen af computer. Herudover er det meste af arbejdet i klubben frivilligt arbejde, hvilket betyder at de enkelte medlemmer, der står for administrationen, kan være der ca. en gang hver anden uge.
Et mål som ”safety” er prioriteret som irrelevant, grundet systemets beskedne størrelse.

3.6.2 Experience

”Helpful ” er prioriteret som meget vigtigt, eftersom brugerne af systemet ikke behøver være fortrolige IT-brugere. Herudover ses det, at målet ”fun and entertaining” er prioriteret som værende irrelevant, eftersom det anses at IT-systemet skal være et seriøst hjælpeværktøj og ikke en form for underholdning.

3.7 Interaktionsformer

Til understøttelse af den begrebsmæssige model, er der en række interaktionsformer, som skal medvirke til en bedre forståelse af interaktionen med systemet. Dette skal hjælpe til at fastlægge, hvorledes et objekt udpeges og hvordan en funktion aktiveres. Ligeledes bliver der også set på, hvordan data indlæses og opstilles [DIEB5]
I denne prototype kan der blandt andet ses en menu-struktur, som hjælper brugeren med at navigere mellem siderne. Endvidere hjælper dette til at organisere informationen, hvilket giver en styret og kontrolleret informations- og funktionsoversigt. Menuen har sine fordele i, at der ikke kræves en stor indlæring for at forstå det, hvilket samtidigt sørger for færre indtastninger.
En anden interaktionsform ses der i form af dialog, hvilket blandt andet forekommer hvis en skytte ikke findes i systemet under en søgning. Dette skal medvirke til, at brugeren ved, hvad systemet gør og ikke er i tvivl, hvis der opstår fejl.
Når der skal indtastes nye resultater, benyttes skemaudfyldelse til at hjælpe brugeren med en hurtigere indtastning. Ligeledes kræver denne interaktionsform en begrænset træning, da skemaet vejleder én igennem indtastningerne.
Den næste interaktionsform er WIMP, hvor der benyttes vinduer som platform. Denne form for interaktion er let at huske, hvilket er vigtigt for denne form for system. Dette skyldes, at det ikke benyttes dagligt og derfor skal være let at anvende.
Ved den sidste form for interaktion er der udskrivning af data, hvilket forekommer i et vindue, f.eks. under udskrivningen af resultater og skytteinformationer.

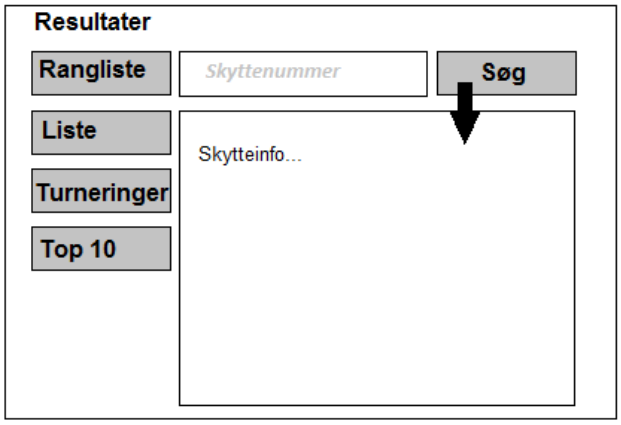
3.8 Designprocessen

Ud fra brugeranalysen er estimeret, at de sandsynlige brugere af systemet som værende ”ikke-erfarne” indenfor IT. Dette har været et væsentligt aspekt under udviklingen af brugergrænsefladen, da dette betyder at brugergrænsefladen skal være så simpel og intuitiv som muligt.
Programmets brugerflade er designes i Windows Forms, hvilket også er fordelagtigt i forhold til aktørerne. Dette skyldes at de, selvom de muligvis ikke har nogen større erfaring indenfor IT, er bekendte med WIMP - brugerfladen fra dagligdags brug.

3.8.1 Indledende tanker

I SAD modulet blev forskellige udkast til grunddesignet skitseret og resultatafviklingen er her inddraget som en startskitse for prototypens brugergrænseflade.

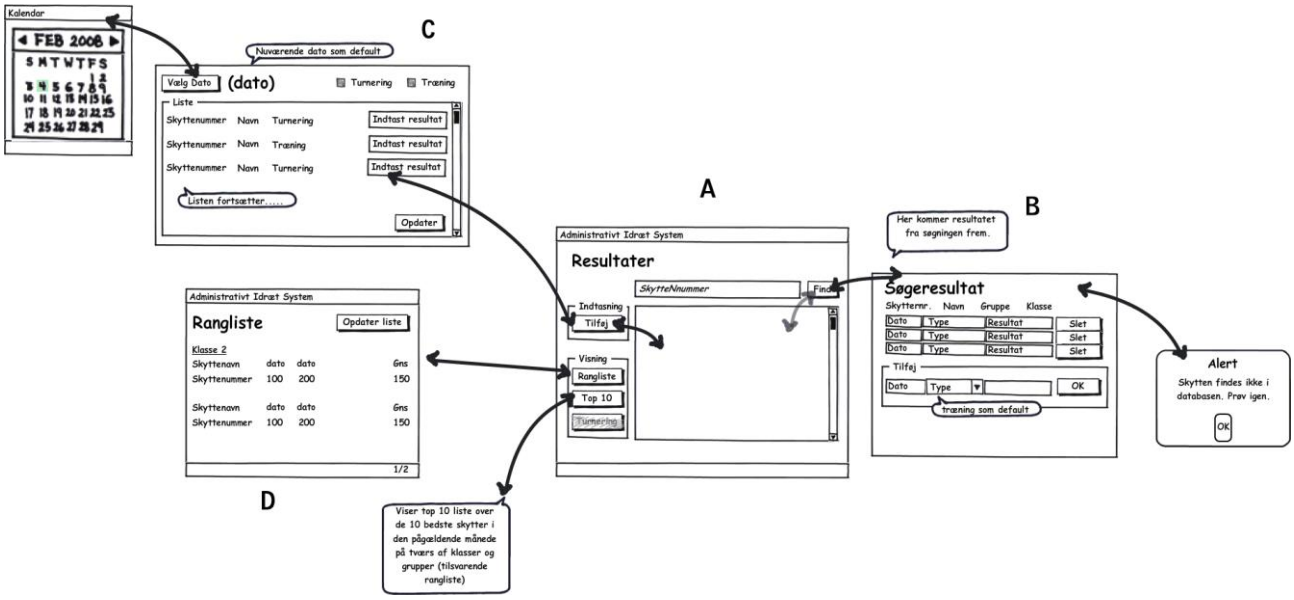
Resultatfremvisningen er opbygget med funktionsknapperne vertikalt placeret til venstre, og med et horisontalt søgefelt med en svag grå markering med navnet ”skyttenummer”, for at sikre, at aktøren ved hvad der skal skrives i feltet. (se figur 20)



Figur 20 - Vindue: Resultater

3.8.2 Mockups

Efter de indledende tanker på hvordan brugerfladen kunne designes, blev der i starten af DIEB-forløbet lavet mockups, som vist nedenunder i figur 21. Formålet er at give et overblik over hvilke nye vinduer brugeren ser, når han/hun aktiverer noget i systemet. De enkelte vinduer er lidt mere detaljeret i forhold til de indledende tanker.



Figur 21 – Mockup (De enkelte ovenstående vinduer kan ses i bilag 2)

Disse billeder blev lavet, som en skitsering af de ideer der var omkring systemets opsætning over for brugeren, hvilket skulle hjælpe til at danne grundlaget under fremstillingen. På den første side (A) er knapperne indkapslet i kategorierne indtastning og visning, hvilket skal medvirke til en bedre forståelse for, hvad knapperne fører til. Systemet har til formål, at hjælpe skytteklubben med resultathåndtering på en sådan måde, at det ikke tager lang tid at indtaste nye data. Dette ses blandt andet ved anvendelse af en kalender, hvorpå der kan udvælges hvilken dato resultatet tilhører. Default-værdien er dags dato. Der kan ligeledes markeres, om resultatet stammer fra en træning eller en turnering, hvorpå der blot skal indtastes resultaterne. Her er default sat til træning, da denne er langt hyppigere indtastet end turneringsresultater. Derudover skal der bemærkes, at der i hvert eneste vindue er en overskrift markeret, så aktøren ved hvor i systemet, han/hun opererer nu.

I systemets design, er der blevet designet to metoder, hvorpå receptionisten eller administratoren kan indtaste nye resultater. Det skyldes, at der kan søges efter den enkelte skytte via skyttenummeret. Hvis der søges på en skytte (Se B), så kan der øverst på skærmen ses, hvilken skytte der er taget fat i, samt hvilken gruppe og klasse skytten tilhører. I den anden tilføjelsesmetode (Se C) kan der vælges en dato, hvorpå en li-

ste af skytter, som er tilmeldt den dato vises. Brugeren kan dermed hurtigt indtaste resultater for flere skytter på én gang på den samme dato. Skulle det ske, at der ikke er tilmeldt nogen skytter på en given dato, så vil en besked blive udskrevet på skærmen, som fortæller aktøren, at der ikke er mulighed for at oprette nogen resultater og hvorfor.
Uanset hvilken metode der vælges, så vil resultatet blive udskrevet på samme vis under resultatfremvisningen. Dette betyder også, at der kan overskrives et resultat, hvis skytten har de samme indtastninger et andet sted. Skulle det ske, at der indtastes et forkert resultat, så kan linjen slettes via en sletteknap og oprette et nyt resultat. I denne prototype vil begge resultatindtastninger finde sted i det samme vindue, som systemet åbner i, hvilket skal bidrage til et bedre overblik, end hvis brugeren pludselig sidder med flere åbne vinduer på én gang og mister overblikket.
Ved resultatfremvisningen findes der to typer, som er henholdsvis en rangliste (se D) og en top ti. Ranglisten har til formål, at vise skytternes placering indenfor deres gruppe og klasse via en timer som skifter side. Dette skyldes, at en skytteklub kan bestå af mange skytter, hvilket ikke kan vises på en skærm alene. Dette vindue bliver ligeledes vist på en ekstern skærm, således at administratoren og receptionisten forsat kan benytte systemet. Ved systemet top ti vises de ti bedste skytter på tværs af grupper og klasse, hvilket har til hensigt at give lidt mere konkurrence blandt medlemmerne i klubben. Begge resultatfremvisninger vil blive vist i et nyt vindue og har en opdateringsknap, som sørger for at opdatere listerne ved ændringer.

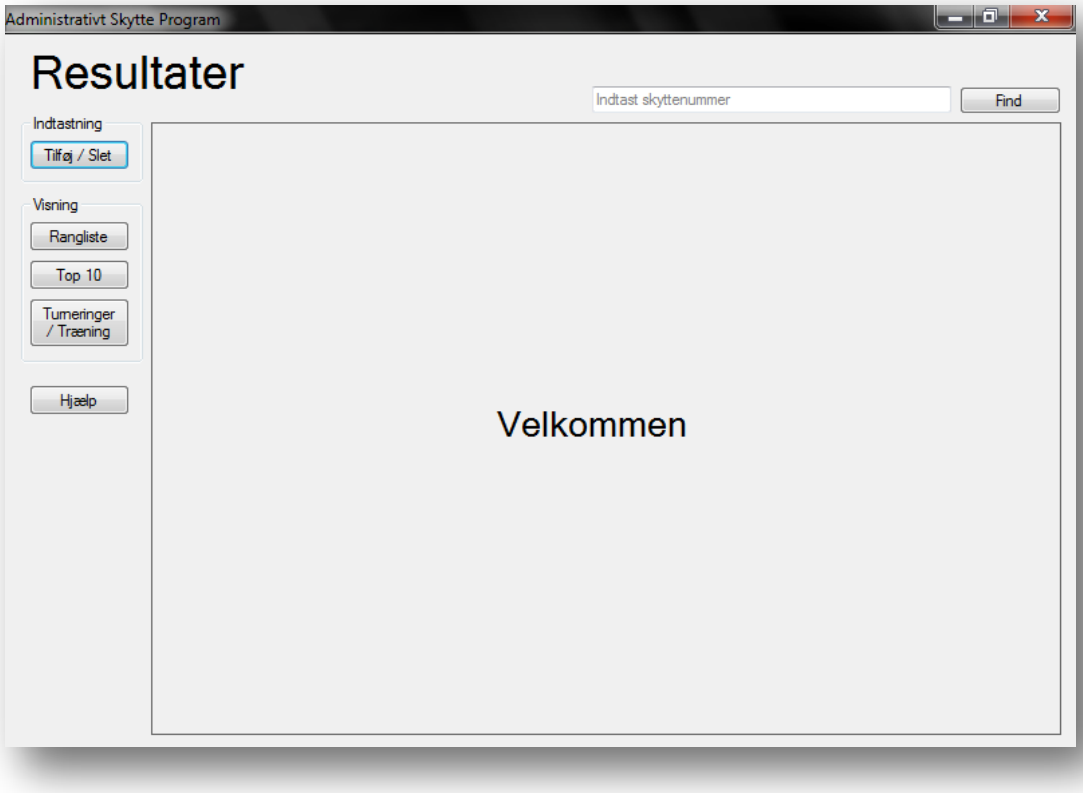
Som der tidligere er blevet nævnt, så er der høj fokus på at systemet skal være let at anvende og huske, da brugeren måske ikke er meget it-bevendt. Dette betyder, at brugeren bliver hjulpet i tilfælde af, at der skulle opstå en fejl. Det kan være, at der bliver indtastet et forkert skyttenummer, hvorpå en alertboks vil blive udskrevet på skærmen og fortælle at der er sket en fejl. På den måde ved receptionisten eller administratoren, hvor fejlen er sket og hvorfor.

Ud fra mockups bliver den endelige brugergrænseflade designet. Billeder af tidligere designs kan findes i bilag 5.

3.8.2.1 Gestalt-lovene

I det endelige design bliver Gestalt-lovene indarbejdet som et vigtigt element i designet. Formålet er at sørge for, at brugeren opfatter programmet som det er tiltænkt og vil finde det intuitivt og tilfredsstillende at bruge. [DIEB6]

Her følger eksempler på hvordan Gestalt-lovene er indarbejdet i programmets brugerflade.

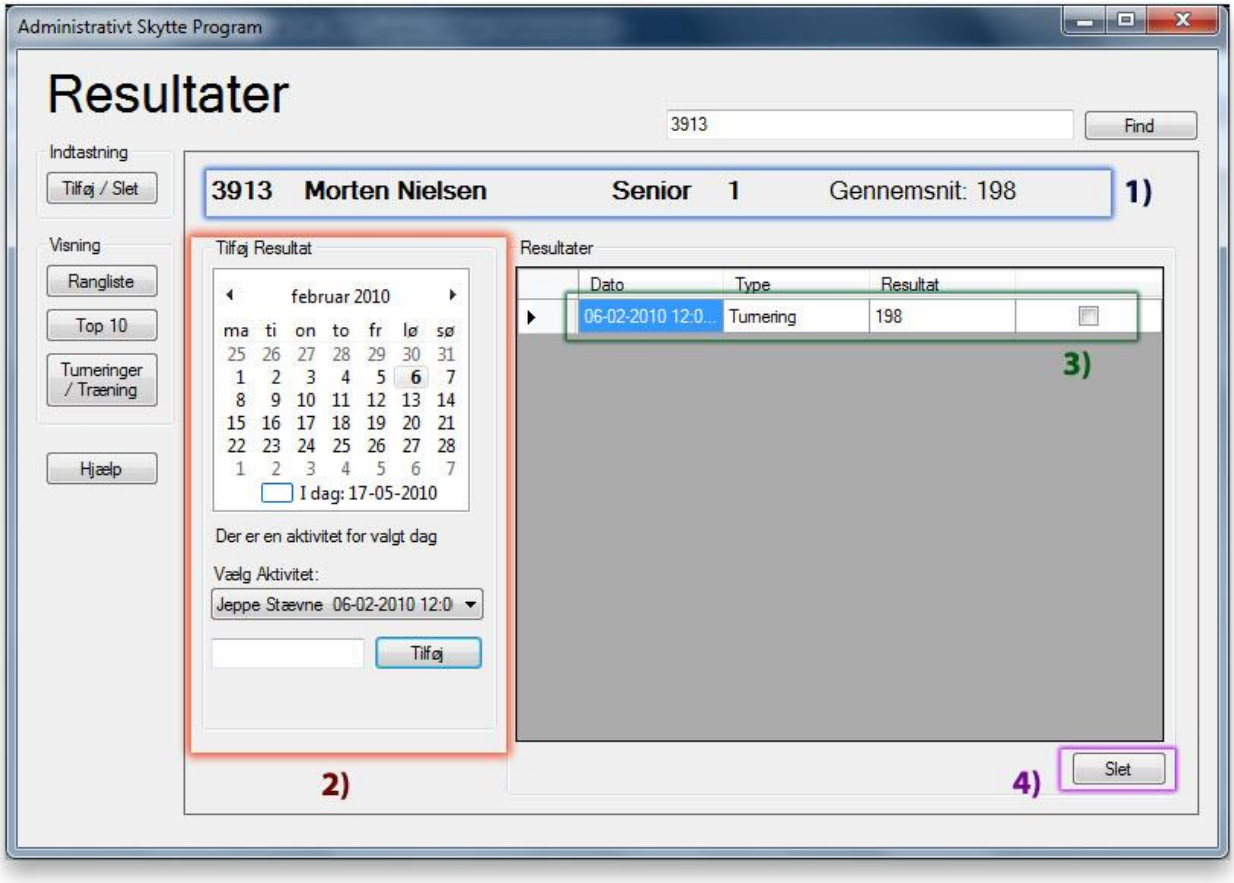


Figur 22 - Resultatside

Figur22 er et screenshot af prototypens resultatside. Her er Gestalt-lovene blandt andet benyttet i knapperne til venstre. Ved at bruge ”loven om lukkethed” er knappen ”Tilføj/Slet” blevet indkapslet i en svagt markeret boks. Dette bevirker, at hændelsen ”indtastning” hører til knappen. Samme regel er gældende for begivenheden visning og de tre tilhørende knapper; Rangliste, Top 10 og Turneringer. Aktøren får derved en intuitiv fornemmelse af, at såfremt han ønsker at indtaste noget skal han trykke på ”Tilføj/Slet”, og hvis han/hun ønsker at få en visning af noget, skal han trykke på den af de tre visningstyper han ønsker at se. Derudover er Gestalt-lovene om nærhed og ensartethed benyttet, for at give indtryk af at alle ”event - knapperne” hører sammen i venstre side af skærmen.

Søgefeltet er sat i øverste højre hjørne af vinduet. Ordene ”Indtast skyttenummer” er svagt markeret med grå sådan at aktøren ikke er i tvivl, om hvad der skal indtastes. Når aktøren markerer i søgefeltet, vil teksten

forsvinde. Gestalt-lovene om nærhed og ensartethed er benyttet her. Dette skyldes, at søgefeltet er linjeret med knappen "find", som er lige ved siden af søgefeltet.



Figur 23 – Resultatside med opstregning

På ovenstående figur 23 ses det hvordan loven om nærhed er implementeret omkring skyttenummer, navn, gruppe og klasse(1), da der er en usynlig linje, som fortæller brugeren at dette hører sammen. Nedenunder findes en lukket boks(2), hvori det er muligt at tilføje et nyt resultat til skytten. Loven om nærhed benyttes også i denne lukkede boks(2), da det kan ses at kalenderen, aktivitetsvalg og resultatindtastningen hører sammen. Det samme tilfælde findes der under resultater, som indkapsles i en lukket boks(3), hvor de forskellige data står listet. Her er der tale om nærhed, da det er bekendt hvilke datoer der eksempeltvist hører til et givent resultat. Ligeledes er der tale om ensartethed, da de forskellige data listes på samme vis hele vejen igennem tabellen. I denne tabel benyttes også en række checkbokse til højre boks(3), hvori aktøren kan afkrydse de linjer, der ønskes slettet via knappen under tabellen boks(4). Da disse elementer ligger på linje med hinanden, kan aktøren se at disse hører sammen.

Igennem hele programmet er der kontinuitet og ensartethed, da systemet er organiseret som en helhed. På hvert vindue er tingene sat op på en sådan måde, at de ligger tæt op af de andre. Dette medvirker til, at brugeren ikke skal tilvænne sig en ny opsætning ved hvert vindue. Endvidere bidrager dette til, at styrke hukommelsen og læringen af programmet. Kontinuiteten medvirker til, at brugeren kan genkende grundfunktionerne i programmet og nemmere kan finde ud af at navigere i programmet.
I forhold til programmets prioriteringstabel er memorability og learnability prioriteret henholdsvis ”meget vigtig” og ”vigtigt”. Derfor er kontinuitet vigtigt i opbygningen af brugergrænsefladen.

3.9 Det endelige design

Under udviklingen af programmet blev der foretaget større ændringer i nogle ting, i forhold til de mockups, der blev tegnet tidligere i projektet (Se figur 22 og bilag 2), for at gøre designet endnu mere brugervenligt. Her kan der nævnes, at når der søges på en skytte, så er det blevet muligt at taste "Enter" i stedet for at trykke på ”Find”-knappen hver gang, der søges på en skytte. Når der er blevet søgt på en skytte, så kan det ses at de omtalte sletteknapper er blevet fjernet ved hvert resultat og er blevet erstattet med selectbokse i stedet for. Dette skyldes, at det er let at komme til at trykke på knapperne ved et uheld, og samtidig sænker det effektiviteten, hvis der ønskes at slette flere resultater på samme tid, da der skal tastes slet ved hver linje.
Tilføjelsesboksen er flyttet til venstre side, hvilket skyldes at det ville være besværligt for brugeren, at skulle ned i bunden af siden hver gang der skal tilføjes et nyt resultat. Dette ses i særdeleshed, hvis programmet kommer til at indeholde mange resultater.
En anden ting, der er blevet ændret i forhold til søgningen, ses i form af alertboksen, som ville komme frem hvis skytten ikke fandtes. Dette er blevet erstattet af en besked på skærmen, som fortæller at skytten ikke eksisterer. Begrundelsen til dette skyldes, at en alertboks kun anvendes ved større fejl, hvorpå brugeren muligvis kan blive utryg ved f.eks. en forkert indtastning.
Der er blevet programmeret regler for at sikre, at aktøren ikke kan indtaste bogstaver i søgefeltet. Ligeledes er det ikke muligt, at have flere visningsvinduer af samme slags åbnet på én gang, hvilket bidrager til en bedre overskuelighed.
Dato-feltet er ændret til en kalender i tilføjelsesvinduet og de dage, hvor der er aktivitet (en turnering eller træning) er markeret med fed skrift. Denne ændring er foretaget for, at gøre det mere overskueligt for brugeren, idet det nu også er muligt at se ugedagene og de dage der er aktivitet, og dermed hjælper det brugeren med at finde en aktivitet hurtigere. Når aktøren vælger en dato, der er markeret med fed skrift, udfyldes der informationer i en dropdown boks, hvor det er muligt at vælge en af dagens aktiviteter.

Når en aktivitet er valgt, opdateres listen således, at de skytter der er tilmeldt denne aktivitet bliver fremvist, hvorefter det er muligt at indtaste et resultat til dem. Det tidligere design indeholdte en lang liste med skytter, der deltog i en hvilken som helst aktivitet på den valgte dato. Men i det nye design er aktiviteterne delt op i tid. Alle de skytter der har tilmeldt sig til for eksempel træning fra klokken 19:00 til 19:30, vil blive vist i en liste og alle de, der har tilmeldt fra 20:00 til 20:30 i en anden liste. Dette gør det nemmere og hurtigere for administratoren at indtaste resultater for skytterne, idet listen ikke er så lang som før og grupperet i den tidsperiode, de er på skydebanen.
De andre elementer på vinduet er flyttet således, at de stadig overholder reglerne om nærhed og følger læseretningen.

3.10 De 4 designprincipper

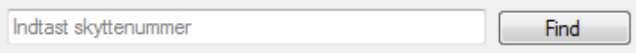
De 4 designprincipper er affordance, mapping consistency og feedback, og disse principper kan hjælpe med at præcisere, hvad der er et godt eller dårligt design. Som systemudvikler er det vigtigt at fokusere på disse principper, da det kan hjælpe til at forstå, hvorledes et godt design opbygges. [DIEB7]
Hvis de menneskelige evner og handlingsvaner ikke implementeres korrekt, kan der forekomme alvorlige designfejl. Derfor er der forsøgt at følge disse 4 interaktionsdesignprincipper under designprocessen, for at få opbygget et godt design.
Herunder følger en gennemgang af alle 4 principper, og hvorvidt det er implementeret i prototypen.

3.10.1 Affordance

Affordance er et begreb som dækker visibility og constraints.
Visibility beskriver, hvordan elementer kan repræsenteres i forhold til hvordan de skal bruges. På denne måde er aktøren ikke i tvivl om, hvordan han skal løse en given opgave. Constraints på den anden side handler om, at forhindre brugeren i at gøre noget, som han ikke bør gøre.
I dette projekt arbejdes der med Windows forms. Da dette er noget langt de fleste har kendskab til, er de færreste i tvivl om for eksempel hvad en knap er, radiobuttons er og hvad en menu er. I programmets brugerflade er der også lagt vægt på enkelhed. Derfor er der ikke for mange informationer på en gang. Knapperne er implementeret tydeligt med en tekst, som der beskriver den event de udfører. Dette fungerer også når aktøren skal indtaste et resultat, her er der tydeligt markeret hvad der skal indtastes og i hvilken rækkefølge.

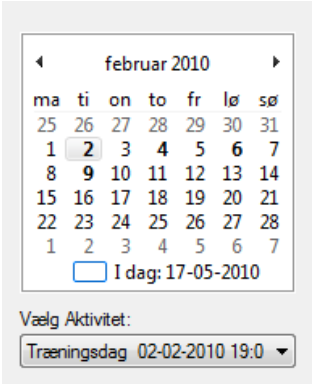
I søgefeltet (figur 24) er der visuelt beskrevet, hvad aktøren skal indtaste og en begrænsning er sat på således, at aktøren ikke kan skrive andet. Et visuelt cue til hvad der skal indtastes i søgefeltet, er den svage grå

tekst pålydende ”Indtast skyttenummer”. Her fremgår det derfor relativt tydeligt, hvad der skal gøres for at søge efter en skytte: Der indtastes et skyttenummer i søgefeltet og der trykkes på knappen find. Grunden til at knappen er døbt med teksten ”Find” og ikke ”Søg” er, at når en aktør ser ”søg” forventes en liste med søgeresultater, mens ”find” forbindes med et enkelt absolut resultat, der ledes efter.



Figur 24 - Søgefelt

Visible constraints er for eksempel blevet anvendt under tilføjelsesfunktionerne, i form af en kalender(figur 25). Her er de dage, hvor der er aktiviteter, markeret med en fed skrift, således det fremgår tydeligt for brugeren, hvilke dage der er aktivitet og hvilke dage der ikke er. Dette er en stor hjælp for brugeren, idet han nemmere kan finde den dag han ønsker, når kun de dage hvor der er aktivitet er fremhævet.



Figur 25 - Kalender

3.10.2 Mapping

Mapping angiver sammenhængen mellem kontroller og deres effekt på det, de styrer. I en direkte mapping er der god sammenhæng mellem kontrollen og det der styres og det vil derfor formindske tænkearbejdet hos brugeren.

Figur 26 - Visning

Dette princip er blandt andet implementeret i funktionsknapperne til venstre i vinduet. Knapperne er markeret i en svagt aftegnet gruppeboks, som bevirker at brugeren opfatter knapperne som en helhed, henholdsvis for aktionen ”indtastning” og for ”visning”. Under visning er det blevet besluttet at gruppere alt, der har med visning at gøre(Figur 26). Disse tre knapper åbner alle et nyt vindue, hvor der vil blive vist resultater uden mulighed for yderligere form for interaktion. Når brugeren først har trykket på én af knapperne, kan han forvente at noget lignende ville ske, hvis der trykkes på de to andre, hvilket er formålet.

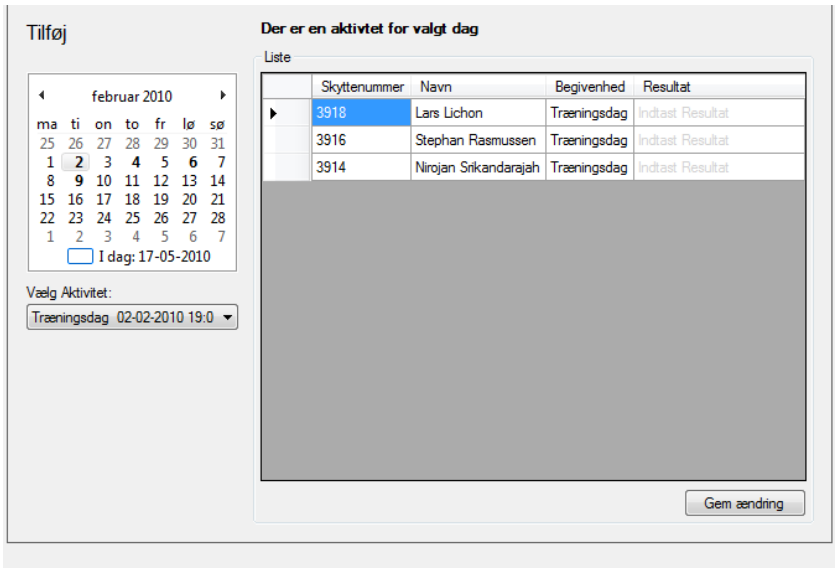
Under tilføjelsesfunktionerne er ”gem” knappen grupperet med listen, således der ikke kan opstå tvivl om, at denne knap gemmer ændringerne, der er foretaget på listen.

Det samme er gældende for søgefeltet, som står lige ved siden af og har den samme højde som knappen ”find”. Derved opfatter aktøren også dette som en helhed.

3.10.3 Consistency

Consistency er når ensartede objekter og funktioner i et interaktivt produkt ser ud og fungerer på samme måde. Dette kan være en god ide, da brugeren derved kan genbruge sin erfaring og derved sparer han tid, ved at lede efter de elementer, han ønsker at benytte.
Dette er implementeret over hele programmet, da alle vinduer er nogenlunde ens bygget op: Aktionsknapper til venstre, søgefelt øverst til højre og informationsvinduet centreret.

Det er ligeledes for at skabe ensartethed, at begge tilføjelsesfunktioner er opbygget med samme struktur (sammenlign figur 27 med figur 22). Det ses, at begge tilføjelsesvinduer består af en kalender til venstre og en liste med resultater til højre. Knappen "Slet" i tilføjelsesvinduet som frembringes ved brug af søgefunktionen (se figur 22) og knappen "Gem" under tilføjelsesvinduet som frembringes ved hjælp af "Tilføj/slet" knappen er placeret det samme sted i vinduet, til højre under resultatlisten. I den første tilfælde kan knappen "Slet" slette flere resultater fra listen på samme tid (hvis flere resultater markeres), og "Gem"-knappen i det andet tilfælde gemmer alle de indtastede og slettede resultater.



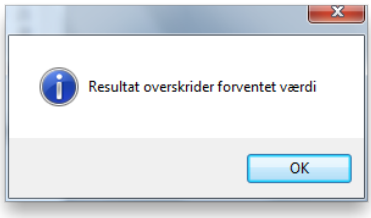
Figur 27 - Tilføj/Slet

3.10.4 Feedback

Feedback fortæller brugeren, hvad der er sket med systemet efter han har udført en handling. Hvis systemet for eksempel er ved at arbejde, skal brugeren have det at vide, så han ikke tror at systemet ikke reagerer på hans handling. Systemet interagerer med brugeren flere steder i programmet. Eksempelvist i søgefeltet, hvor der står "Indtast skyttenummer". Såfremt brugeren indtaster et ikke-eksisterende skyttenummer så prompter programmet, brugeren med teksten "Skyttenummer findes ikke". Det samme vil ske såfremt brugeren indtaster andet end et tal i søgefeltet, dog med teksten "Indtast et skyttenummer".

I tilføjelsesvinduet med en skyttes resultater (figur 22), vil der komme en dialogboks som spørger om man er sikker på at slette, hvis der trykkes på "slet" knappen. Denne interaktion er vigtig for at undgå, at der bliver slettet noget ved en fejl.
I tilføjelsesvinduet med aktiviteter (figur 27) har teksten "Indtast resultat" i resultatfelterne en grå farve. Når der indtastes resultater i felterne, forbliver farven stadig grå; den bliver først sort når der klikkes på "Gem"-knappen. Dette fortæller brugeren, hvornår de indtastede resultat er gemt.

For at håndtere fejlhandlinger og indtastninger fra brugerens side, er der opstillet regler for hvornår der skal dukke en meddelelsesboks op, der har til formål at informere brugeren om fejlen. Et eksempel på sådan en meddelelsesboks ses på figuren nedenunder (figur 28).



Figur 28- Meddelelsesboks

I det følgende vil alle de fejlbeskeder, som skal dukke op i form af meddelelsesbokse under fejlhandlinger og fejlindtastninger blive beskrevet.

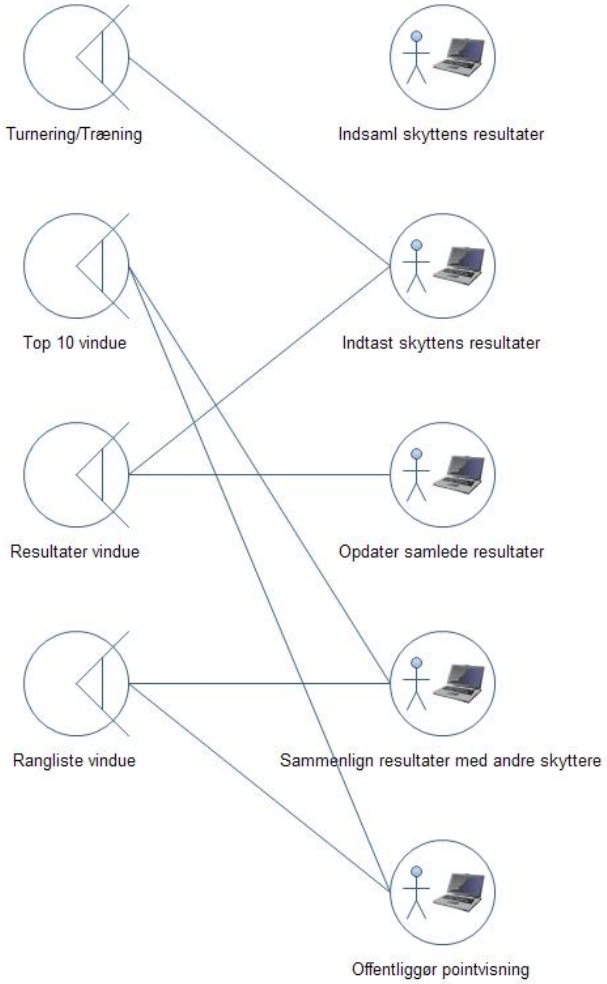
Under tilføjelsesvinduet, hvis der indtastes et resultat der er større end 200 skal brugeren få beskeden: "Resultatet overskrider forventet værdi".
Hvis der vælges en dato, der ligger senere i kalenderen end dagens dato, skal brugeren få beskeden:
"Du kan ikke indtaste resultat for fremtidige aktivitet"
Ligeledes skal brugeren ikke kunne indtaste bogstaver eller andet tegn end tal i resultatfelterne, ellers skal beskeden "ikke et gyldigt resultat" dukke op.

3.11 Interaktionsrum – En analyse af brugergrænsefladen

Interaktionsrum giver en visuel fremstilling af programmets interaktion med aktørerne på et abstrakt niveau [DIEB4]. Under udviklingen af programmet, er der ikke blevet brugt interaktionsrum, da brugerfladen var lavet efter inspiration fra modellerne tegnet i SAD-kurset. Derfor bliver interaktionsrummene lavet og benyttet lige inden brugerfladen er færdiggjort, for at lave en sammenlignende analyse. For at få et overblik over de forskellige interaktionsrum, som er knyttet til brugergrænsefladen, opstilles der to interaktionsrum i alt, for de forskellige aktører, som blev udarbejdet i SAD casen. I prototypen af systemet, er der således tale om fire aktører, nemlig administrator, receptionist, skytte og tilskuer. De har hver især adgang til forskellige dele af systemet.

Den færdige brugergrænseflade er ikke nøjagtig magen til den brugergrænseflade der blev udarbejdet i SAD-casen. Dette skyldes at der kun er blevet fokuseret på resultater, fordi der kun bliver lavet en mindre del af hele systemet. Af denne grund er der kun tre aktører i stedet for oprindeligt fire aktører, da intet af det som receptionistens job indeholder, er med i prototypen, som f.eks. oprettelse af skytte og til – og afmelding af bane, turnering og træning samt da skytterne og tilskuernes interaktion sammen med systemet er den samme, er de slået sammen som en.

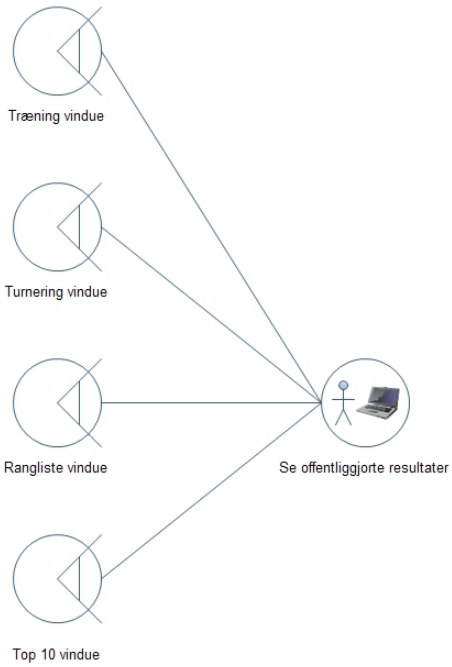
På figur 29 er der vist interaktionsrum for administrator og på figur 30 er der vist interaktionsrum for skytte/tilskuer.



Figur 29 - Interaktionsrum for administrator

Det er kun administratoren, der har rettighederne til at tilføje, slette, gemme og ændre informationer i systemet, hvorfor interaktionsvinduet ser sådan ud. Skytten og tilskueren kan kun se de forskellige resultater.

Som systemet er opbygget passer det sammen med interaktionsrummene. Som udarbejdet i SAD-casen ville det kun være administrator, der kan interagere direkte med systemet og indtaste, ændre osv. Tilskuerne kan så aflæse stillinger, resultater, ranglister mm. Derfor viser analysen, at interaktionsrummene passer godt med systemet, selvom de ikke var direkte inkorporeret under udviklingen.



Figur 30 - Interaktionsrum for skytte/tilskuer

3.12 Usabilitytest

Systemet er nu kommet til den fase, hvori det skal testes overfor en bruger. Dette er medvirkende til at give et indblik i om programmet fungerer efter hensigten, eller om der er problemer under interaktionen. Testen er en lille usability test, da der bliver arbejdet med en demo af et endeligt system, hvorpå der er begrænsninger i forhold til, hvor meget der kan blive testet. Den valgte testform til testen bliver en assesment-test.

Testen foregår hjemme hos en af gruppens medlemmer, som er bekendt med testpersonen. Selvom testlederen kender testpersonen vil der stadig være en introduktion til testen.

Testens roller vil bestå af en testleder og en datalogger. Testlederen har til formål, at sørge for at testen går som den skal. Han hjælper ikke direkte testpersonen, hvis han eller hun skulle gå i stå under selve testen. Testlederen sørger ligeledes for at minde testpersonen om at tænke højt, således at dataloggeren kan notere alle tanker omkring anvendelsen af systemet. Dataloggeren har til formål at notere alle relevante observationer, som opstår under interaktionen. Disse noter uddybes således, at dataudviklerne kan rette de problemer der end måtte være ved systemet. Endvidere vil der efter testen blive foretaget en række efter-sessionsspørgsmål, hvor testpersonen har mulighed for at give sin overordnede mening til kende, i forhold-til interaktionen.

Før testen forventes det, at få et indblik i om brugerens tanker stemmer overens med den måde, systemet er opbygget på. Materialer til usabilitytesten kan ses i bilag 3.

3.12.1 Testens resultater

I tabellen nedenfor ses en tabel over de fundne problemer i usabilitytesten

Problem nr.	**Beskrivelse**	**Problemkategori**	**Tidsforbrug**
Opgave 1			1:38
P1	Benytter ikke søgefunktionen til at løse opgaven.	Kosmetisk - brugeren løser opgaven uden problemer.	
Opgave 2			2:16
P2	Brugeren undrer sig over at der ikke kan tastes Enter for at gemme.	Kosmetisk - bruger kan løse opgaven ved at benytte musen til at taste gem.	
Opgave 3			1:02
P3	Brugeren undrer sig over at der ikke kan tastes Escape for at lukke vinduet ned.	Kosmetisk - brugeren kan lukke vinduet ned ved at trykke på kryds.	
Opgave 4			1:29
P4	Brugeren undrer sig over at der ikke kan trykkes frem og tilbage mellem siderne i ranglisten	Kosmetisk - dette foregår på ekstern skærm og vil derfor styre sig selv.	
Opgave 5			1:15
P5	Benytter ikke søgefunktionen til at løse opgaven.	Kosmetisk - brugeren løser opgaven uden problemer.	
P6	Ved ikke hvordan man sletter.	Alvorlig- brugeren løser opgaven ved at erstatte resultatet i stedet for at slette, gemme og derefter oprette nyt.	
Opgave 6			1:12
P7	Bliver overrasket over at der er et slideshow og ønsker mere kontrol.	Kosmetisk - brugeren kan godt se ideen med slideshowet og løser hurtigt opgaven.	
Opgave 8			1:05
P8	Benytter ikke søgefunktionen til at løse opgaven.	Kosmetisk - brugeren løser opgaven uden problemer.	
			I alt 12:14

Tabel 11 – Tabel for testens resultater

3.12.2 Eftersessionsspørgsmål

Efter interviewet blev der foretaget nogle eftersessionsspørgsmål. Testpersonen fandt programmet logisk opbygget til formålet. Dog ønskede han, at det kunne styres, hvilken side i de fremviste slides det ønskedes at være på. Han mener dertil også, at det kunne være smart, hvis brugeren kunne indtaste hvor mange sekunder der skulle gå i mellem hver slide, da han kender et andet system med denne funktion. Dog finder han hastigheden til at være passende.
Ligeledes mente han, at tastaturet skal kunne bruges mere til at udfylde resultater, da det bliver til gene at benytte musen hver gang. Dette nedsænker ligeledes effektiviteten. I top 10'en blev der savnet, at der kunne ses hvilken klasse og gruppe skytten befandt sig i, men finder generelt denne del som en god idé, da dette er en sjov lille konkurrence til skytteforeningen.
Brugeren kunne godt lide, at det var neutrale farver, og mener at der er en god forklaring rundt om i systemet. Messageboksene som kommer frem ved fejl, finder han passende, så der ikke kan indtastes noget vrøvl. Brugeren kunne også godt lide at teksten blev sort, når der blev gemt under tilføj/slet. Menuen er godt placeret, da der bliver læst fra venstre mod højre. Han mener derfor at søgefeltet burde flyttes hen til menuen. Han følte dog ikke han havde behovet for at benytte denne funktion, ud fra de stillede opgaver. Ligeledes ville han ønske, at der kunne søges efter navn. Aktøren stillede også spørgsmålstegn ved de lyde, som opstod ved indtastninger og lignende, da denne lyd normalvis betyder advarsel.
Testpersonen savner ligeledes, at der kan ændres direkte på linjen under fundet skytte, i stedet for at skulle slette først og derefter tilføje. Denne side blev ikke benyttet under selve testen, hvorpå der blev opdaget en række mindre problemer under eftersessionen, som ikke ville have været opdaget uden.

3.12.3 IDA-session

Usability-resultaterne blev gennemgået via IDA metoden. De fundne fejl blev diskuteret og revurderet ud fra et nyt synspunkt [DIEB2].
Følgende fejl fik ændret karakter ud fra IDA-sessionen:
P2 & P3 (problemer vedrørende manglende funktioner på tasterne ”Enter” og ”Escape”) er blevet ændret fra kosmetisk til alvorlige problemer. Problemerne vil over en længere periode medføre frustration hos brugeren, da programmet på nuværende tidspunkt kræver aktive klik på musen for at udføre eventet.
P4 og P7 (problemer vedrørende slideshowfunktion i programmet) er blevet ændret fra kosmetisk til alvorlig, da det er frustrerende for brugeren ikke at kunne få de informationer, der ønskes med det samme. Aktøren er på nuværende tidspunkt tvunget til at afvente, at det ønskede slide kommer frem af sig selv. Da testen kørte, var der kun to sider med slides, hvilket medførte at der ikke var så lang ventetid. I en rigtig

brugssituation med mange flere skytter, klasser og grupper vil ventetiden være lang, hvilket er ensbetydende med frustration.

P6 (problem vedrørende "slet"-funktionen) fik ændret karakter fra alvorlig til kosmetisk, da testpersonen fuldførte opgaven hurtigt, dog ikke som planlagt i testen. Testpersonen erklærede, at han ikke vidste hvordan han skulle slette, hvorfor han bare direkte erstattede det indtastede i stedet. Dette kan også skyldes, at der i testen ikke blev stillet en opgave, hvor testpersonen udelukkende skulle slette. Det er udviklernes mening, at testpersonen ville have løst opgaven korrekt, såfremt han var blevet stillet denne opgave i stedet, da han var i den rigtige del af programmet.

Problem nr.	Oplevet antal	Beskrivelse	Problemkategori
P1	3	Benytter ikke søgefunktionen til at løse opgaven, hvilket var hensigten.	Kosmetisk
P2	1	Der kan ikke tastes Enter for at gemme.	Alvorlig
P3	1	Der ikke kan tastes Escape for at lukke vinduet ned.	Alvorlig
P4	2	Brugeren undrer sig over at der ikke kan trykkes frem og tilbage mellem siderne i ranglisten	Alvorligt
P5	1	Erstatter i stedet for at slette og derefter tilføje.	Kosmetisk
P6	1	Visningen af Turnering/Træning fremstår som et diasshow.	Kosmetisk.

Tabel 12 – Tabel over problemer efter endt IDA-session

3.12.4 Fejlkilder

Følgende fejlkilder kan have påvirket testresultaterne.

- Usabilitytesten blev kun foretaget på én person
- Testen blev foretaget i et miljø, hvor testpersonen ikke var hjemmevant
- De til testpersonen stillede opgaver kan være misforståede i forhold til hensigten
- Spørgsmålene kunne også være struktureret på en anden måde, så testpersonen ikke vidste hvor han kunne finde eksempelvis gennemsnit under top 10'en i stedet for under find.

3.12.5 Refleksion over testen

Ud fra den færdige usabilitytest, blev der reflekteret over resultater og testforløbet. Der blev blandt andet fundet diverse små fejl i testen, som blev rettet i prototypen. I en eventuel fremtidig version, skulle alle fundene fejl rettes. Gennemgående ønskede brugeren mere kontrol over systemet. For at give dette, skulle brugeren have flere funktionsaktiverende taster til programmet, så vedkommende selv kunne bestemme hvor han vil være i eksempelvist listen. Et andet element, der skulle ændres var, at der skulle bindes aktiviteter til eksempelvist ”escape” knappen og ”enter” knappen.

3.13 Delkonklusion

Målet med DIEB-forløbet var at konstruere en brugergrænseflade der er brugervenlig, ud fra de lærte principper. I den forbindelse blev designet af brugergrænsefladen fra SAD-afsnittet brugt som en skabelon, hvorefter det blev revideret, således at designet overholdte designprincipperne og gestaltlovene, for at øge brugervenligheden. Der er blevet lavet en brugeranalyse i form af stakeholders og personas, hvor der blev lavet en biografi af en fiktiv person, der kunne være en bruger af systemet. Denne person var derfor i tankerne mens designet blev revideret, således at designet bedst muligt blev målrettet mod denne person. Under forløbet har der været større diskussioner, og der blev ændret i designet flere gange. Valget blev truffet for at overholde designprincipperne, og for at designe en brugergrænseflade, der kunne hjælpe aktørerne med at spare så meget tid som muligt.

Der er blevet foretaget en usability evaluering, efter den metode der blev anbefalet af DIEB-kursusvejlederen. Et medlem fra skytteklubben var testpersonen og der blev efter revidering og vurdering i alt fundet seks usability problemer, hvor tre af dem kan kategoriseres som alvorlige og tre som kosmetiske.

Testen viste generelt, at der i prototypen mangler kontrol til aktøren, som bruger programmet. Dette skal rettes til en eventuel næste version.

4 Implementering

Kapitlet beskriver arbejdet med at programmere prototypen, som bliver programmeret i programmeringssproget C#. Ud fra det, der er blevet gennemgået i de tidligere cases, er der blevet udarbejdet en prototype af systemet.

I Objekt-Orienteret Programmering og Algoritmik (OOPA) kurset, er der blevet undervist i de forskellige emner indenfor Objekt-orienteret Programmering. På 1. semester indeholdte programmeringskurset det mest grundlæggende indenfor programmering i C#, og der blev programmeret meget imperativt. I dette projekt er undervisningen i OOPA blevet benyttet til at programmere systemet objekt-orienteret. Systemet er udformet i Windows Forms. Der henvises til kursusholder Kurt Nørgaards hjemmeside, da denne er blevet brugt i forbindelse med projektarbejdet. [OOPA9]

Formålet med OOPA er, at kunne udvikle et simpelt objekt-orienteret IT system, som ikke er trivielt, og det skal kunne anvendes til at varetage forskellige administrative opgaver. Det er derfor også et mål, at kunne arbejde meget objektorienteret.

4.1 Metode

Programmeringsmodulet vil bære betydeligt præg af ændringer, og heri vægtes en korrekt og præcis kode-repræsentation. I SAD og DIEB modulet fremgår det tydeligt, hvorledes systemets opbygning af objekter, klasser, attributter og metoder skal repræsenteres i programmet. Det er også på baggrund af disse analyser og designs, at systemet tager sin grundform.

I objektorienteret programmering opdeles koden i objekter, og der findes tre grundprincipper, som er indkapsling, nedarving og polymorfi. Indkapsling betyder, at et objekt indkapsler nogle data og noget opførsel. Nedarving betyder, at et objekt kan nedarve data og opførsel fra et andet objekt, dvs. at én klasse i programmet kan nedarve fra en anden klasse. Polymorfi betyder så, at programmet bliver brudt op i objekter, da de enkelte dele af programmet så kan isoleres fra hinanden.

To af disse principper er blevet brugt i dette projekt, indkapsling og nedarving.

4.1.1 Test af program

Der er under dette projekt blevet foretaget en række test af programmet, for at sikre at programmet virkede efter hensigten. Testning af et program er ressourcekrævende og det menes, at der ca. anvendes 40 % af udviklingstiden til tests[OOPA1]. En god test siges at være, når der er en høj mulighed for at finde fejl. Program testning kan udelukkende vise nuværende fejl, hvorved det er muligt at skrive testen før programmeringen påbegyndes. I den virkelige verden skal alle kombinationer og metoder testes igennem, så det er fuldstændigt sikkert, at der ikke opstår nogen uventet fejl. Selv i mindre programmer, er der et stort antal muligheder.
Der findes forskellige former for tests, hvoraf der kan nævnes black box og white box test. I disse test findes der flere niveauer. Der er her tale om enhedstest, integrations test og system test. Hvorvidt et program er test-bart refereres til kvaliteten af softwaren, som påvirker evnen til at finde fejl. Her kan nævnes fire typer, hvilket er henholdsvis observability, kontrollerbarhed, decomposability og forståelighed. Observability er resultatet af udskrivningen, som skal være synlig. Kontrollerbarheden er der, hvor der er fokus på inputtet og stadie til programmet, således at der hele tiden er kontrol før testen. Decomposability er den testform, hvor programmet deles op i dele og testes individuelt. Forståeligheden er specifikationen af programmet, hvor der skal dannes en forståelse for, hvad der er korrekt adfærd i koden.

4.1.2 White box test

I en white box test, anvendes der kontrolstrukturer af en program enhed, som kan udlede forskellige test cases [OOPA1]. Formålet med denne test er derfor at sikre, at alle kommandoer og udtryk i et program udføres mindst én gang. Ligeledes skal alle logiske afdelinger, loops og datastrukturer udøves på deres grænser. Stien igennem en sådan test, forgår på den måde, at der vælges et minimalt sæt af stier i programmet, så alle udsagn og betingelser bliver dækket.
Udførslen af denne test går derfor ud på, at tegne et flow chart(rutediagram) af program enheder. Dette flow-chart skal dernæst laves til et rutediagram, hvorefter cyclomatic komplekset bliver fundet (såsom n) – test-metric. Til sidst skal n identificeres igennem flere test cases, hvor der følges n's rute.

4.1.3 Black box test

En black box test benytter et interface fra en programenhed, til at udlede en række test cases [OOPA1]. Dette foregår således under selve systemudviklingen, hvor programmet bliver systematisk testet for eventuelle fejl. Ligeledes betyder det, at der foretages en ny test ved hver ændring, så det sikres sig at programmet virker efter hensigten. Formålet med denne test er, at demonstrere om programenhederne producerer det forventede output ud fra et egnet input. Der ønskes derfor at lokalisere funktionelle fejl, inter-

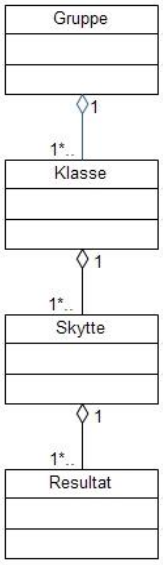
face fejl og effektivitets fejl. På den måde findes der frem til, om der opnås det forventede resultat ud fra et givent input, og om data bliver leveret korrekt til og fra andre metoder. Ligeledes opnås der viden, om programmet arbejder hurtigt, eller om der er grundlag til forbedringer.
Når der laves en black box test, så er det ikke realistisk at prøve alle kombinationer af mulige inputs i programmet. Det er derfor nødvendigt at træffe et valg af input værdier, hvorpå de vigtigste bliver prøvet af. Det er derfor også en god idé at afprøve input, som på forhånd regnes for at være invalide, hvorpå der kan elimineres de fejl, som måtte opstå i sådanne situationer. Ligeledes skal der inkluderes grænse repræsentanter, såsom tomme collections og fulde collections.
Det er vigtigt, at få identificeret de forskellige enheder og hvilke der testes. Ligeledes er det vigtigt, at holde øje med, hvilke enheder der udleder en til flere test cases. En enhed kan for eksempel være en pakke af navnerum og forsamlinger, typer i form af klasser og konstruktorer, enlige kommandoer og udtryk, medlemmer af typer i form af procedurer, funktioner og metoder. I et objekt-orienteret program vil de forskellige test enheder ses i form af de individuelle public operationer i hver klasse. Derfor bliver hver enkelt del i programmet testet, isoleret fra de andre dele.
Til at udføre denne test, anvendes NUnit, som er et enheds testing program. NUnit har til egenskab at markere stykker i testprogrammet via særlige attributter, hvorved korrektheden angives ved brug af assertions, som er metoder med et boolean resultat (sandt eller falsk). Programmet tilbyder en lang stribe af forskellige attributter og assertions, som kan hjælpe med at teste. NUnit værktøjet er ligeledes i stand til at markere den angivne test i en eller flere samlinger, hvor det i dette tilfælde ses som en samling i et dll fil format. Dette program kan anvendes til at teste, både ved hjælp af en console runner eller en GUI runner.

4.2 Afgrænsning

Der er i denne case blevet besluttet at afgrænse sig fra, at analysere på begge skydediscipliner, hvorved der udelukkende fokuseres på 15 m. Dette skyldes at koden til luft vil minde meget om koden til 15 meter, hvorpå det ikke findes interessant at gentage noget der allerede er lavet. Ligeledes vil der ikke blive arbejdet med gruppe- og klassefordeling, på en sådan måde at brugeren vil blive promptet med en besked om flytning, i det denne del af programmet foregår i medlemsvinduet, som blev fravalgt i DIEB afsnittet. Der afgrænses således også til at kigge på hold, grundet særlige regler og betingelser. Reglerne for hold er blandt andet, at skytterne på tværs af klasser kan tilmelde sig en turnering, og da der ikke arbejdes med booking, vil dette udlede til større programstruktur.

4.3 Ændringer i forhold til klassediagram

Helt i begyndelsen af dette projekt, blev der arbejdet med i alt 20 klasser. Disse 20 klasser blev fundet igennem analysen (jævnfør kapitel 2). Det blev dog senere bestemt, at kun at ligge vægt på en bestemt del af systemet, hvilket var den del af systemet, der har med resultat behandling at gøre. Denne beslutning gjorde blandt andet, at mange af klasserne kunne skæres fra, og klasserne som udviklingsholdet endte med var henholdsvis gruppe, klasse, skytte og resultat. På figur 31 ses et klassediagram over konstruktionen, efter disse ændringer blev foretaget.



Figur 31 - Klassediagram ved implementeringsstart

4.4 Ændringer i programmet

4.4.1 Omstrukturering af klasser

I starten af programmeringsforløbet var programmet struktureret på følgende måde (dette er kun udklip for at vise, hvordan tankegangen var og det er kun enkelte metoder og instans variabler der er vist):

```
class Gruppe
{       private string gruppeNavn;
        List<Klasse> klasser = new List<Klasse>();

        public void AddResultat(int klasseNr, int skytten, DateTime dato, int resultat, int type)
        {
            this.klasser[klasseNr].AddResultat(skytten, dato, resultat, type);
        }
….

class Klasse
{       private int klasseNr;
        private List<Skytte> skytter = new List<Skytte>();

        public void AddResultat(int skytten, DateTime dato, int resultat, int type)
        {
            this.skytter[skytten].AddResultat(dato, resultat, type);
        }
….
```

```
class Skytte
{       private int skytteNr;
        private string navn;
        private DateTime fDato;
        private List<Resultat> resultater = new List<Resultat>();

      ….

class Resultat
{       int resultat;
        int type;
        DateTime dato;

        public void AddResultat(DateTime dato, int resultat, int type)
        {
            this.resultater.Add(new Resultat(dato, resultat, type));
        }
```

Ovenstående er et hierarkimønster, hvor gruppeklassen indeholdte en liste med klasser af typen klasse, og denne klasse "klasse" indeholdte en liste med skytter af typen skytte, og inde i skytteklassen er der en liste med resultater af typen resultat. Denne opbygning blev valgt idet det så ud til, at være den mest logiske måde og da den stemte overens med klassediagrammet fra analysedelen. Men det var ikke var den mest praktiske og effektive måde, at strukturere programmet på. Hver gang der skulle hentes data fra de underliggende klasser, skulle der køres igennem hver klasse for til sidst at komme til den klasse, hvor den ønskede information lå. Hvis det for eksempel ønskes at finde et resultat for en skytte, skulle gruppeklassen benyttes og derefter ind i den rigtige liste. Herfra kommer benyttes klassen "klasse", hvor der tages stilling til hvilken klasse skytten ligger i.

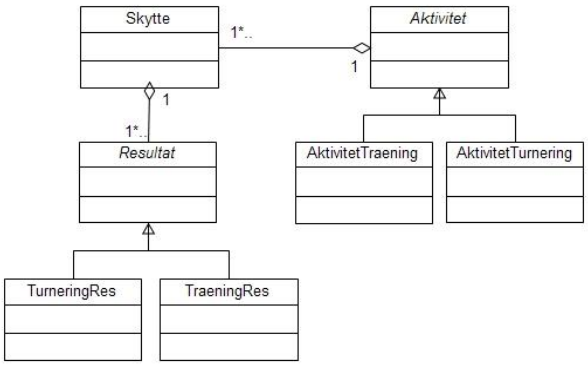
Dernæst entreres skytteklassen, hvor der skal tages endelig stilling til, hvilket resultat på resultatlisten der ønskes. Alt dette blev gjort ved hjælp af metoder, der sørgede for at kommunikere med de andre klasser. Dette ene problem ved denne form for struktur var, at hver gang skulle der tages stilling til hvilken gruppe og klasse en skytte tilhørte, når der ønskedes informationer om skytten. Det andet problem var, at det var kompliceret, når en skytte skulle flyttes ind i en anden klasse eller gruppe. For at simplificere dette blev der, efter vejledning fra OOPA-kursusvejlederen, valgt en anden programstruktur.

Både gruppe- og klasseklassen blev nu fjernet. Da gruppeinddeling sker på baggrund af en skyttes alder og klasse inddeling af skyttens gennemsnit, var det ikke nødvendigt at komplicere processen med at holde styr på hvilken gruppe og klasse en skytte tilhørte, og inddele skytterne i grupper fra starten. Gruppenavnet og klassenummeret kunne altid findes, når det ønskes ved hjælp af skyttens alder og gennemsnit, ved at lave metoder der sørger for at give gruppenavn og klassen efter alder og gennemsnit. Dog var det ikke så simpelt, da det ikke altid var muligt at bestemme gruppenavnet ud fra alderen. Hvis en skytte er med i gruppen ungdom eller senior, er der et specielt tilfælde, hvor der er mulighed for at vælge om skytten vil være i den

tildelte gruppe eller den anden gruppe, som har navnet ”Åben gruppe”. Da der nu er tale om gruppeinddeling med valg, og ikke en gruppeinddeling, der kan ske fuldstædig automatisk på baggrund af informationer, kan gruppe og klasseinddeling ikke kun bestemmes ud fra metoder. Det valg der er foretaget, skal huskes og gemmes. Derfor er der nået frem til, at klassen skytte skulle have en instansvariabel ved navn gruppenavn af typen string, som husker, hvilken gruppe en skytte er med i. Herudover blev der lavet 5 ny klasser ”Aktivitet”, ”AktivitetTraening”, ”AktivitetTurnering”, ”TurneringRes” og ”TraeningRes”.

De tre aktivitetsklasser blev oprettet på grund af hardcoding, da der ikke arbejdes med booking delen af programmet, hvor skytten tilmelder sig en træning eller turnering. De har således til formål at finde ud af, hvilke skytter der er tilmeldt hvilken træning på hvilken dag . AktivitetTraening og AktivitetTurnering er ligeledes underklasser til Aktivitets klassen (Dette ses der også nærmere på i afsnit 4.4.2 Hardcode).

I klassediagrammet, som blev beskrevet i SAD afsnittet, var der en træning- og turneringsklasse, hvor resultatklasserne arvede fra disse fordi resultaterne skulle opdeles. Dette er der blevet lavet om på i programmet, i det der er fokus på resultat. Det vil sige, at nu arver TraeningsRes og TurneringsRes fra resultatklassen, så der på den måde kan tjekkes hvorvidt skyttens resultat befinder sig i en træning eller en turnering på en given dato. På figur 32, ses det hvordan ændringer har indflydelse på formen af det nye klassediagram.



Figur 32 - Klassediagram efter programændringer

4.4.2 Hardcode

I denne prototype befinder der sig en smule hardcoding, som skal demonstrere de dele af programmet, som blev fravalgt under udviklingen. Der er blandt andet tale om medlemsvinduet, hvori det skulle være muligt for receptionisten at tilføje og administrere klubbens skytter. Dette blev hardcodet ind i programmet, således at skytten fik indskrevet navn, fødselsdato og skyttenummer. Et eksempel på dette kan ses nedenfor:

```
skytter.Add(new Skytte(3913, "Morten Nielsen", (new DateTime(1985, 06, 16))));
```

Disse skytter blev ligeledes tilmeldt til række aktiviteter, hvor nogle befandt sig på forskellige tidspunkter samme dag. Denne del stammer oprindeligt fra bookingvinduet, hvor skytten skulle tilmelde sig enten en træning eller en turnering. Et eksempel på denne hardcode kan ses nedenfor:

Her oprettes selve aktiviteterne:

```
aktiviteter.Add(new AktivitetTraening(new DateTime(2010, 05, 12, 18, 00, 00)));
aktiviteter.Add(new AktivitetTurnering("Jeppe Stævne", new DateTime(2010, 02, 06, 12, 00,00)));
```

Her tilføjes skytterne til en aktivitet:

```
aktiviteter[2].TilføjSkytte(skytter[5], skytter[3], skytter[1]);
```

4.4.3 Programklassen som en postkasse

At implementere overgangen fra ”Console Application” til ”Windows Forms Application”, var en mere tidskrævende proces end forventet. Det administrative system som console-application, var færdig i et forholdsvis tidligt stadie, idet de nødvendige klasser, og de vigtigste metoder var blevet programmeret. Måden der kunne hentes data fra forskellige klasser og den eneste måde hvorpå klasserne kunne kommunikere med hinanden, var ved hjælp af metoder og properties. Hver gang en information fra en anden klasse skulle bruges, blev der kaldt på diverse metoder og properties. Der blev derfor brugt for meget tid og ressourcer på at hente data på tværs af klasserne. Det medførte til, at der blev valgt at lave en public static klasse, som kunne fungere som en slags postkasse, som indeholdte vigtige data, som de andre klasser kunne få adgang til direkte uden at kalde nogen metode.

4.5 Metoder i programmet

I dette afsnit ses 4 uddrag af kode fra systemet, dette projekt bygger over. Alle eksemplerne er metoder i diverse klasser, og hver metode spiller hver deres vigtige rolle i forhold til programmet og dets gøren. Afsnittet er lavet, for at give læseren et overblik over det kode, der er blevet fundet mest relevant eller interessant.

4.5.1 Metode: LoadResultater (i klassen Skytte)

Denne metode har en utrolig relevant rolle i programmet, da den sørger for at enten at oprette .dat filer (i tilfælde af, at der er indtastet en ny skytte i systemet), eller åbner allerede oprettet .dat filer for allerede oprettet skytter (Der henvises til figur 33).

```
205 public void LoadResultater(int input)
206 {
207     string Liste = input.ToString();
208     string file = Liste + ".dat";
209     if (!File.Exists(file))
210     {
211         using (FileStream strm =
212         new FileStream(Liste + ".dat", FileMode.Create))
213         {
214             IFormatter fmt = new BinaryFormatter();
215             fmt.Serialize(strm, resultater);
216         }
217     }
218     else
219     {
220         using (FileStream strm =
221         new FileStream(Liste + ".dat", FileMode.Open))
222         {
223             IFormatter fmt = new BinaryFormatter();
224             resultater = fmt.Deserialize(strm) as List<Resultat>;
225         }
226     }
227 }
```

Figur 33 – Kodeeksempel (Screenshot)

Linie 205: På denne linje annonceres det, at denne metode er public, da andre klasser skal kunne få fat i den, samt den er af typen void, da den ikke skal returnere nogen værdi. Herefter ses det, at metoden har en parameter af typen int som kaldes ”input”. Dette input svarer til skyttens placering på listen, altså den givende skyttes index på listen.

Linie 207: Her oprettes der en string med navnet ”Liste”. Den bliver tildelt værdien af inputtet, som er en givende skyttes index placering (fx ”1”). Inputtet konverteres til string, da det er af typen int.

Linie 208: Her oprettes der en string med navnet ”file”. Den bliver tildelt værdien af det, der står på Liste, samt strengen ”.dat”. (fx ”1.dat”).

Linie 209: En If-else betingelse starter her. Den bruges til at tjekke, om en fil med navnet af værdien af det der står på file (fx ”1.dat”) allerede eksisterer. Hvis filen ikke eksisterer, vil programmet oprette en ny fil, men hvis filen derimod eksisterer, vil programmet åbne den givende fil.

Linie 211 - 212: Hvis overstående tilfælde ikke var sandt, vil programmet nu oprette en ny fil for den givende skytte. Programmet laver her en FileStream af navnet ”strm” (FileStream benyttes til at skrive og læse data af fx typen string). Strm skal være lig med en ny FileStrem med navnet Liste + ”.dat” (fx ”1.dat”) og FileMode sættes til Create, da der som sagt skal oprettes en fil og ikke åbnes en fil.

Linie 214: Der oprettes et interface af typen IFormatter kaldet ”fmt”. Dette bliver sat lig med en ny Binary-Formatter, som er formatteringsklasse.

Linie 215: fmt sættes til at Serialize (informationer skrives ud på en fil i et binært format). Informationer tages fra skyttens resultat liste og lægges ud op FileStream ”strm”.

Linie 220 – 225: Det der forgår på disse linjer, er stort set det samme som fra til og med linje 211 – 215. Dog med små forskelle. For det første ses det, at der ikke er tale om FileMode.Create, men om FileMode.Open. Grunden til dette er, at hvis filen allerede eksisterede, skulle programmet blot åbne den. Herudover ses det på linje 214, at skyttens resultater skal sættes lig med det ”Deserialized” produkt af FileStream strm, som en liste af typen Resultat (Resultat er en klasse i programmet).

4.5.2 Metoder til gruppeinddeling (i klassen Skytte)

Disse metoder sørger for, at inddele den enkelte skytte i den rigtige gruppe, baseret på skyttens alder. Dette har en særlig relevans i forhold til visninger i forbindelse med ranglister, turneringer og træning (Der henvises til figur 34).

```
53  #region Gruppe- og klasseinddeling
54  public void GruppeInddeling()
55  {
56      int alder = this.GetAlder;
57      if (!(gruppeNavn == "Åben"))
58      {
59          if (alder <= 14)
60          { this.gruppeNavn = "Børne"; }
61          else if (alder >= 15 && alder <= 17)
62          { this.gruppeNavn = "Junior"; }
63          else if (alder >= 18 && alder <= 20)
64          { this.gruppeNavn = "Ungdom"; }
65          else if (21 <= alder && 55 >= alder)
66          { this.gruppeNavn = "Senior"; }
67          else if (alder >= 56)
68          { this.gruppeNavn = "Veteran"; }
69      }
70  }
71
72  public void ÅbenGruppe()
73  {
74      int alder = this.GetAlder;
75      if (alder >= 18 && alder <= 55)
76      { this.gruppeNavn = "Åben"; }
77  }
78
79  public void ResetÅbenGruppe()
80  {
81      this.gruppeNavn = "";
82  }
```

Figur 34 – Kodeeksempel (screenshot)

Metoden ”GruppeIndeeling”:

Linie 54: på denne linje annonceres det, at denne metode er publi,c da andre klasser skal kunne få fat i den, samt den er af typen void, da den ikke skal returnere nogen værdi. Herudover er metoden parameterløs.

Linie 56: Her oprettes en instansvariabel af typen Int med navnet ”alder”, som tildeles værdien af den skytte, der arbejdes på alder.

Linie 57: En If-else betingelse starter her. Den bruges til at sørge for, at det næste stykke kode ikke skal køres igennem, hvis skytten har gruppe navnet: ”Åben”.

Linie 59-68: Hvis skytten ikke var i gruppe Åben, vil skytten blive tildelt en gruppe, baseret på skyttens alder. Fx hvis en skytter er under eller er 14 år, så vil skytten komme i Børne gruppen eller hvis skytten er over eller er 56 år, vil skytten komme i Veteran gruppen.

Metoden ”ÅbenGruppe”:

Linie 72: På denne linje annonceres det, at denne metode er public da andre klasser skal kunne få fat i den samt den er af typen void, da den ikke skal returnere nogen værdi. Herudover er metoden parameterløs.

Linie 74: (se Linie 56).

Linie 75-76: Hvis skytten er mellem 18 og 55 år vil skytten blive tildelt Åben gruppen.

Metoden ”ResetÅbenGruppe”:

Linie 79: På denne linje annonceres det at, denne metode er public, da andre klasser skal kunne få fat i den, samt den er af typen void, da den ikke skal returnere nogen værdi. Herudover er metoden parameterløs.

Linie 81: Her sættes skyttens gruppenavn lig med en tom streng.

Sammenspillet mellem disse klasser:

Grunden til, at der bruges disse 3 metoder til at holde styr på tildelingen af en gruppe, er fordi at der er krydsninger mellem alder i forhold til hvilke grupper skytter kan være i. Blandt andet kan man ved alderen 19 både være i gruppen Ungdom eller i gruppen Åben.

Disse metoder er dog ikke blevet brugt helt efter hensigten, da medlems administration er skåret fra i prototypen af programmet. Men ideen var, at når programmet starter vil en skytte blive tildelt en gruppe. Dog hvis en gruppe manuelt er blevet lagt over i gruppen Åben, så skal programmet ikke automatisk ligge skytten for eksempel over i gruppen Ungdom eller Senior, uden at brugeren manuelt selv går ind og foretager denne ændring.

4.5.3 Metode: SetMarkedDates (i klassen Find)

Denne metode har til formål at markere de datoer i kalenderen, hvor der eksisterer en aktivitet, så det er tydeligt for brugeren, hvornår der finder en aktivitet sted (Der henvises til figur 35).

```
312 public void SetMarkedDates()
313 {
314     findKalendar.RemoveAllBoldedDates();
315     for (int i = 0; i < Program.aktiviteter.Count; i++)
316     {
317         for (int j = 0; j < Program.aktiviteter[i].GetSkytterTilmeldt.Count; j++)
318         {
319             if (Program.aktiviteter[i].GetSkytterTilmeldt[j].GetNavn == Program.skytter[skytteIndex].GetNavn)
320             {
321                 findKalendar.AddBoldedDate(Program.aktiviteter[i].GetDato);
322             }
323         }
324     }
325     findKalendar.UpdateBoldedDates();
326 }
```

Figur 35 – Kodeeksempel (screenshot)

Linie 308: På denne linje gøres det klart, at denne metode er public. Dette skyldes at andre klasser skal kunne få fat i den. Denne metode er af typen void, fordi den ikke skal kunne returnere en værdi. SetMarkedDates er navnet på metoden.

Linie 311 - 313: Her ses en for-løkke, som starter med at tælle hvor mange aktiviteter der er. Dernæst tæller den op hvor mange skyttere der er tilmeldt de forskellige aktiviteter.

Linie 315 - 322: Her ses en if betingelse. Hvis der er en skytte tilmeldt en aktivitet, hentes skyttens navn frem, og på den dato, hvor aktiviteten finder sted, tilføjes datoen med fed skrift i kalenderen. Til sidst opdateres de markerede datoer i kalenderen.

4.5.4 Metode: Sorter (i klassen Skytte)

Denne metode har til formål at sortere skytternes resultater på den måde, at de ti bedste resultater står øverst på eksempeltvist ranglisten (Der henvises til figur 36).

```
140 public List<Resultat> Sorter()
141 {
142     List<Resultat> midlertidig = resultater;
143     int antalRes = 0;
144     if (midlertidig.Count > 10)
145     {
146         antalRes = 10;
147     }
148     else
149     {
150         antalRes = midlertidig.Count;
151     }
152     midlertidig.Sort(delegate(Resultat A, Resultat B) { return A.GetResultat.CompareTo(B.GetResultat); });
153     midlertidig.Reverse();
154     int rest = midlertidig.Count - antalRes;
155     midlertidig.RemoveRange(antalRes, rest);
156     return midlertidig;
157 }
```

Figur 36 – Kodeeksempel (screenshot)

Linie 145: Her annonceres metoden sorter til at være public, da andre klasser og metoder skal kunne benytte denne sortering. Ligeledes refereres der til en liste, som indeholder informationer fra klassen Resultat. Denne metode er parameterløs, da metoden arbejder på den enkelte skytte.
Linie 147: Her oprettes der en ny liste kaldet midlertidig, som er lig med resultater, da de oprindelige resultater ikke ønskes redigeret under udskrivningen af de ti bedste resultater.
Dernæst laves der en variabel antalRes, som sættes til at have værdien 0. Denne værdi benyttes nedenfor i en if-else betingelse, hvor programmet finder frem til om skytten har flere resultater end 10. Dette gøres ved hjælp af midlertidig.Count, som tæller op hvor mange resultater skytten har på sin liste. Hvis skytten har over 10 resultater, vil antalRes få værdien 10 og ellers vil antalRes være lig med det antal resultater, skytten har.
Linie 157: Her sorteres listen via en sort metode, som findes i C#. Denne metode opererer ud fra en delegate, der opskriver de betingelser, der er for sorteringen. Der laves derfor en CompareTo metode, hvor et resultat A bliver sammenlignet med et resultat B og derefter returnerer sorteringen. Måden hvorpå der hives fat i resultaterne, ses i form af propetien i resultatklassen, hvor der tages fat i hvert enkelt resultat på listen.
Linie 158: Laves der en reverse metode, som sørger for at listen udskrives med det bedste resultat øverst.
Linie 159: Oprette en ny variabel kaldet rest af værdien int, da der arbejdes med hele tal. Denne variabel sætte til at være lig med midlertidig.Count - antalRes, da der kun skal udskrives de resultater skytten har skudt. Der bliver på linje 160 fjernet overskuddet mellem k og rest, hvorefter listen midlertidig bliver returneret til udskrivning.

4.6 Test af prototypen

I dette projekt er black box testen blevet foretaget efter, at programmet var færdiggjort, hvilket ikke var hensigten, da dette bør gøres undervejs. Denne test bliver derfor benyttet til at validere, om prototypen nu gør det der forventes.
Med Black Box test, blev det valgt at lave en test, der strækker sig over fire klasser. Disse klasser er ”Skytte”, ”Resultat”, ”TræningsRes” og ”TurneringsRes”. Disse 4 klasser blev valgt, da disse klasser har en vigtig rolle i prototypen. Hvis der er fejl i disse klasser, vil disse fejl have indflydelse i resten af prototypen for eksempel i form af, at vise et forkert output til brugeren af prototypen. Følgende skema giver en oversigt over, hvad klassen ”skytte” indeholder:

Klassen:	**Skytte**

Variable	`private int skytteNr;`
	`private string navn;`
	`private DateTime fDato;`
	`private string gruppeNavn;`
	`private int klasseNr;`
	`public List<Resultat> resultater = new List<Resultat>();`

Konstruktor	Klassen har en konstruktor, der sætter værdien af ”skytteNr”, ”navn” og ”fDato”. (denne konstruktor bruges når der oprettes en skytte i systemet)
Properties	Klassen indeholder properties, med Get funktion for alle variablerne samt en property med en Get funktion for skyttens alder.

Metoder	Klassen indeholder følgende metoder:
	`public int UdregnAlder(DateTime birthDate)`
	(udregner skyttens alder baseret på skyttens fDato)
	`public void GruppeInddeling()`
	(inddeler en skytte i en gruppe baseret på skyttens alder)
	`public void KlasseInddeling()`
	(inddeler en skytte i en klasse baseret på skyttens resultat gennemsnit samt gruppe navn)
	`public List<Resultat> Sorter()`
	(Finder skyttens optil 10 bedste resultater og returnere disse med bedste resultater øverst)
	`public double Gennemsnit()`
	(Udregner skyttens gennemsnit baseret på skyttens resultater)
	`public double Top10()`
	(Returner skyttens gennemsnit)

Tabel 13 – Tabel over klassen skytte

Følgende tabel forklarer kort om resten af klasserne:

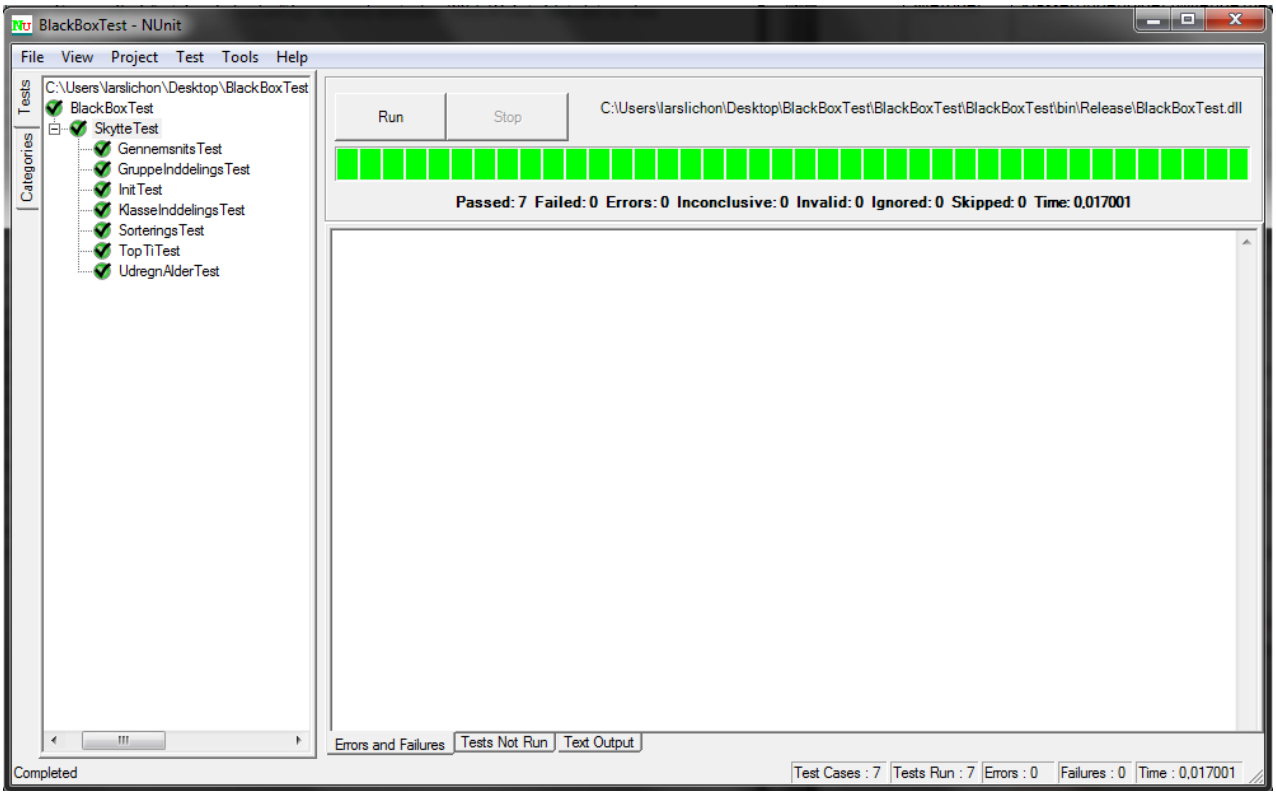
Klassen:	**Beskrivelse:**
Resultat	Abstrakt overklasse, med properties med Get funktion.
TræningsRes	Subklasse der arver fra ”Resultat”, med konstruktor og properties med Get funktion
TurneringsRes	Subklasse der arver fra ”Resultat”, med konstruktor og properties med Get funktion

Tabel 14 – Tabel over resten af klasserne

Grunden til at klasse ”Skytte” har en mere detaljeret beskrivelse, er fordi det er denne klasse der har den største rolle i forbindelse med testen.

I test programmet blev test klassen kaldt ”SkytteTest”. I testen gennemgås alle metoder og properties, for at sørge for, at programmet gør det, som der regnes med.

4.6.1.1 Resultat af testen:



Figur 37 – Nunit Screenshot

Hvis der ses på billedet på siden før (Figur 37), ses det at resultatet af testen var en succes uden nogen fejl. Under testforløbet blev der også lavet decideret fejl, for at finde ud af om testen ville opfange disse og da disse fejl blev fundet måtte det kunne konkluderes at testen virkede.

```
[Test]
public void GennemsnitsTest()
{
        Assert.AreEqual(190, s1.Gennemsnit(), "s1 GennemsnitsTest");
        Assert.AreEqual(170, s2.Gennemsnit(), "s2 GennemsnitsTest");
        Assert.AreEqual(0, s3.Gennemsnit(), "s3 GennemsnitsTest");
}
```

figur 38 – Testkodeeksempel

I figur 38 ovenfor ses et uddrag af test koden. I dette eksempel ses det, at der testes om det udregnede gennemsnit passer til det, som der regnes med. (fx fik skytte s1 resultaterne 180 og 200, hvilket svarer til et gennemsnit på 190).

4.7 Delkonklusion

Gennem en længere proces med analyse, design og programmering er der blevet anvendt en lang række omfattende og mindre omfattende metoder. Programmering af et mindre administrationssystem har givet udfordringer, store som små.
Projektet var grundlag til større overvejelser omkring design, og med en omfattende analyse af programmet, var det med en solid struktur, at programmet kunne tage form gennem den egentlige programmering. Det viste sig tidligt, at der være et stykke fra teori til praksis.

Med justeringer af diverse klasser og mockups undervejs, begyndte anvendelsen af indkapsling og nedarvning at vise sig anvendeligt. Nedarving har givet programmet stor fleksibilitet, muligheden ved at anvende metoder på tværs af klasser har været potentielt for programmets funktionalitet. Ligeledes har indkapsling gjort programmet mere overskueligt og mindre retningsbestemt. Efter en videre implementering af indkapsling, har programmet været væsentlig nemmere at modificere, da det løbene viste sig relevant, at fortage ændringer i bestanddele af programmet.
Et overordnet billede dannede sig via objekternes repræsentationer og da interaktionen dem imellem blev klargjort. Attributter og metoder blev tilknyttet relevante klasser og nedarving blev anvendt til at fremhæve de fundamentale karakteristika herved.
Afslutningsvis er der blevet fortaget en omfattende black box test af programmet, det er herefter sikret at eventuelle uforudsete fejl og mangler i administrationssystemet er rettet.

5 Diskussion

Undervejs i projektforløbet er der sket en løbende udvikling af forståelsen, vedrørende fremstillingen af et IT-system. Forståelsen kom som en naturlig konsekvens af de problemstillinger, som blev bearbejdet undervejs i forløbet. Problemstillingerne var undervejs emne til større diskussion, hvoraf de større er listet nedenfor, opdelt efter de tre caseforløb og afsluttet med en generel diskussion.

Analyse og design

Casens formål var at strukturere tilgangen til systemudviklingen, men de mange visualiseringer i form af diagrammer var svære at overskue i starten, dog efterhånden som arbejdet skred frem, kom overblikket. Forståelsen af de mange nye begrebers betydning, blev afhjulpet af illustrationerne.
Forståelsen for hvem de enkelte aktører var, og at de kunne være én og samme person, gav anledning til diskussioner i gruppen. Skildringen mellem de enkeltes rettigheder og behov i forhold til programmet var vanskelige at have overblik over, da personer i skytteklubben både kan være skytte, tilskuer, administrator og receptionist på én og samme gang.

Designfasen på dette tidlige stadie af udviklingsprocessen var vanskelig at udforme. Det var en udfordring, at lave grundlæggende skabeloner for hvordan brugergrænsefladen skulle designes, da der endnu ikke var undervist i brugergrænsefladedesign. Små vinduer blev konstrueret med en simpel opbygning og uden mange detaljer. Denne mindre designfase blev diskuteret, da designet kunne laves på mange forskellige måder, eksempelvist i forhold til placering af knapper og visning af de nødvendige informationer til aktøren. Undervejs i løbet af planlægningen af programmets udformning, blev diagrammerne flere gange ændret, da nye strukturformer viste sig. Disse ændringer af diagrammerne blev ved under hele projektforløbet, i takt med, at programmets brugergrænseflade skulle implementeres og evalueres. En interessant tilgang til denne udfordring kunne være, at have lavet forskellige designforslag i form af tegnede mockups og derefter adspørge aktører fra skydeklubben, hvad de umiddelbart fandt mest naturligt.

Da der skulle vælges, hvilken del af programmet der skulle arbejdes med, blev resultathåndtering valgt. Undervejs i denne udviklingsproces var der flere forbehold, som skulle påtænkes inden det endelige valg blev truffet. Det var dog svært, at danne overblik over hvor besværligt de enkelte grene af systemet ville være at udvikle, selvom der var lavet illustrationer over hele systemet. Dertil manglede der programmeringserfaring, for at kunne danne et retmæssigt overblik over de enkelte grenes sværhedsgrader. Resultat-

udviklingen blev derfor valgt ud fra, at det virkede interessant, skydeklubben kunne bruge det, samt at det var en matematisk tilgang til udvikling, som gruppen fandt udfordrende.

Brugergrænseflade

Overgangen fra den mere analytiske indgangsfase til den mere konkrete og håndgribelige var vanskelig, da det i starten mindede meget om hinanden. Opfattelsen af udformningen af programmet blev ændret her i forhold til brugergrænsefladen, selvom den valgte del af programmet forblev det samme.
Metoderne, som bruges i brugergrænsefladedesign, skulle implementeres i programmet, hvilket ændrede det simple design konstrueret i analysedelen. Udfordringen undervejs i denne fase var, at fastholde aktørens synspunkt og ikke udviklerens. Dertil hjalp personasanalysen sammenholdt med stakeholderanalysen fra den analytiske del.

Det kan diskuteres, om det var hensigtsmæssigt at usabilitytesten kun blev foretaget med én testperson, da det kan have påvirket testresultaterne og der kan have kommet flere usability problemer frem, hvis der havde været mere end én testperson. De til respondenten stillede opgaver kan have været formuleret på en sådan måde, at de ikke blev opfattet hensigtsmæssigt, hvilket også kan have påvirket resultatet.
Usabilitytestens formål, udformning og forventede udbytte blev diskuteret med henblik på, hvordan den skulle foretages. Der var på forhånd kun kendskab til de tests, som blev undervist i Usability og Usabilityevaluering af IT-Systemer (som blev undervist i, i efterårs semesteret 2009), hvilket krævede stor planlægning. Undervejs i forløbet blev IDA-metoden introduceret. Overgangen mellem disse metoder var vanskelig at danne et overblik over, da den ene krævede mange timers planlægning og den anden var ikke så ressourcekrævende. Dette sammen med det forventede udbytte blev diskuteret ivrigt, da de forskellige former for usabilitytests kræver forskellige resurser og giver forskelligt udbytte. Der kunne have været lavet tests med flere testpersoner i et mere naturligt miljø, og med en mere dybdegående videoanalyse. IDA-metoden blev dog valgt, set i forhold til, hvad testen skulle bruges til.

Der var flere fejlkilder under testen, og der blev diskuteret hvordan de kunne undgås. En pilottest kunne muligvis have hjulpet på formuleringen af spørgsmålene, og flere respondenter til testen kunne have sikret større chance for flere fejl blev fundet. Fejlkilderne kunne have været minimeret ved, at bruge flere testpersoner, da der herved også kan ses om flere af fejlene er gennemgående, og derved kan der bedre vurderes, hvorvidt fejlene er alvorlige eller ej.

Implementering

Overgangen fra imperativ programmering til objekt-orienteret var udfordrende. Da programmeringen blev igangsat, var der meget fokus på den forskel, og resultatet blev en forkert struktur i programmet. Denne forkerte struktur blev forstærket i systemets analysedel, som var meget klasse-orienteret. Løsningen var færre klasser og en anden metodestruktur. Det er essentielt at forstå, hvilke ting som bliver analyseret til at være klasser og hvad der praktisk behøver være det.
En større revurdering af analysedelen bevirkede, at nogle af de valgte klasser i analysedelen blev implementeret, som lister i selve programmet. Denne kobling mellem klasser i analysedelen og hvad der konkret skal være i programmet, viste sig som en større udfordring. Forståelsen for hvad aktøren ser på skærmen og hvad vedkommende tror der sker, i forhold til hvad programmet rent praktisk gør, blev opnået undervejs i denne proces.

Koblingen mellem programmeringen og analysedelen var også udfordrende, i forhold til navigationen i programmet for aktøren. Dette blev afhjulpet undervejs i brugergrænsefladedelen via et navigationsdiagram, som hjalp med overblik. I det hele taget blev der nogle gange programmeret uden at tænke på aktørens rolle i systemet, hvilket er nødvendigt for fremstillingen af et tilfredsstillende program.

Selve udformningen af programmet i forhold til brugergrænseflade, blev også revurderet undervejs. Der blev valgt at benytte data-grids til fremvisning af resultater, da det opstillingsmæssigt er logisk. Dog er det en problematisk visningsform med mange skytter og resultatvisninger. Alternative visningsformer blev diskuteret, men resultatet blev ikke ændret. Der kunne her have været testet forskellige visningsformer, som nævnt i analyse og designafsnittet, for at vide hvad aktørerne foretrækker som visningsform.
Usabilitytesten viste også frustration hos testpersonen over manglende kontrol og visningen af resultaterne, hvilket igen skyldes den valgte visningsmetode.

Generel diskussion

I starten af udviklingsprocessen blev den generelle struktur i systemet diskuteret i forhold til visningsformer for aktørerne. Aktørernes roller var blevet fastlagt og deres visningsbehov klarlagt, men selve repræsentationen for aktøren var ikke fastlåst endnu. Forskellige former for visninger, såsom flere eksterne skærme, slideshow-model, skærme med individuelle styringsmuligheder mm, blev diskuteret, men valget faldt på én ekstern skærm til skytter og tilskuere. Mulighederne for de andre former for visning, kunne have været undersøgt, med hensyn til aktørernes præferencer.

I prioriteringstabellen blev der lagt fokusområder for det udviklede program. Her blev begrebet ”fun” prioriteret lavt under udviklingen, da der var mere fokus på effektiviteten. Dog blev der programmeret en ”Top10” funktion i programmet, netop for morskab og konkurrencens skyld. Det blev diskuteret om denne skulle med, når nu det ikke var et prioriteringsområde, men usabilitytestens testperson var positiv for denne funktion. Flere funktioner af denne type kunne muligvis være lavet, til fordel for aktørerne.

For at mindske arbejdsbyrden for receptionisten og administratoren, kunne der være undersøgt i, hvordan man kunne udvikle programmet til selv, at få skytternes resultater direkte efter skydningen var tilendebragt. Dette ville dog være for kompliceret på nuværende stadie, da det ville kræve elektroniske skydeskive og en systemgrænseflade, der er blevet afgrænset fra.

Efter den første opdeling af klasse-strukturen over systemet, var det vanskeligt at forstå konsekvenserne ved ændringer i strukturen, som de fremkom undervejs. De blev anset for at være for statiske i forhold til det, de nu skulle benyttes til. Denne problemstilling kom under alle tre caseforløb, hvilket betød at der nogle gange kom ændringer i klassestrukturen, mens der andre gange ikke var nogen ændringer.

Overordnet set var der i starten for meget fokus på regler og struktur i opbygningen, hvor der var fokus på at rette sig efter klassediagrammet, i stedet for at rette dette til i takt med udviklingsprocessen blev mere specifik. De tre caseforløb påvirkede på hver sin måde prototypens endelige udseende og funktionalitet, men det kunne have været interessant at have aktørerne mere aktivt med i udviklingsprocessen, end der var.

6 Konklusion

Gennem undervisningen og det løbende arbejde med de tre caseforløb, er vores viden og kunnen vokset med opgaven. Ved brug af omfattende analyse, evaluering og implementering står problemformuleringen til at besvare. Vi har undervejs, stået overfor problemstillinger, som krævede en objektiv og kreativ tankegang, og denne kreativitet er kommet til udtryk i denne rapport. Det at udvikle et program fra bunden, er en udfordrende proces, især for første gangs udviklere. Det er nu muligt at besvare problemformuleringen:

- Hvilke processer er der under udviklingen af et administrativt it system, og hvordan skal dette system opbygges så det dækker skytteklubbens behov på en brugervenlig måde?

Igennem systemanalyse og designkurset, er der blevet undervist i en række metoder, som skal hjælpe til analysen af systemudviklingen. Dette har bidraget til, at der opstilles en række retningslinjer for hvorledes systemet skal designes. Der er blandt andet tale om fastlæggelse af de basale krav til systemet, hvilket skaber grundstenene til den videre struktur. Der er også skabt en større forståelse for, hvorledes et system inddeles i klasser, således at grundstrukturen kommer på plads. På den vis kan man som systemudvikler se, hvorledes de forskellige klasser skaber forbindelse til hinanden.
I SAD-kursusforløbet blev der også undervist i anvendelsesområdet, hvorved der lægges fokus på aktørerne for at finde frem til, hvem de er og hvorledes de interagerer med systemet. På den vis kan man som systemudvikler få et bedre indblik i, hvordan aktørerne tænker, så programmet kan designes til dem. Dette gjorde, at de første skitser af brugergrænsefladen kunne tegnes op med tilhørende forklaring til funktioner. Derpå kunne et navigationsdiagram laves, så der skabes overblik over, hvilke knapper der fører til hvad, og hvilken forbindelse der er mellem vinduerne. Igennem dette caseforløb, er der blevet arbejdet iterativt i den forstand, at modeller og metoder har måttet revurderes ud fra ny viden senere i forløbet. Kurset havde afslutningsvis lidt udarbejdelse af design og arkitektur, hvorved der i dette projekt blev dannet en prioriteringsliste og en komponentliste med tilhørende skema. Der er på den vis også opnået en større viden og forståelse for, hvorledes en case tilrettelægges til at tilpasse sig det projekt, som man sidder og arbejder med.
Der blev skabt et grundlag for at arbejde videre med designdelen af projektet, via DIEB kurset. Ud fra skitseringen fra systemanalyse og designkurset, blev der arbejdet videre ud fra en række designprincipper. Med et større fokus på brugeren, blev programmet målrettet til vedkommendes behov. Der har under dette forløb været megen diskussion om, hvorledes designet skulle omstruktureres. Valgene blev derfor truffet

ud fra de designprincipper, der er blevet undervist i, som kan hjælpe aktøren i at løse sine opgaver mere effektivt og tilfredsstillende.

Der blev undervist i en ny metode at foretage usability evaluering, som ikke kræver at der benyttes et laboratorium for at afprøve sit program. Under denne test blev der fundet en række overraskende problemer, som kunne rettes og forbedre programmet yderligere ved brug i skytteforeningen. Dog dækkede programmet de overordnede krav og opgaverne blev løst i løbet af kort tid, hvorpå det virker som tiltænkt. Programmet kan derfor karakteriseres til at være godt til formålet, men der kan dog foretages en række forbedringer, som sikrer mod fremtidig frustration ved et større antal indtastninger og lignende.

Denne kan således konkluderes som en god test, da vi opdagede fejl, som vi ikke ville have set uden testen. Ligeledes gjorde IDA sessionen, at vi fik en mere præcis bedømmelse af problematikkerne.

I det sidste case forløb er der blevet undervist i, hvorledes man programmerer objektorienteret. Dette har bidraget til at udarbejde selve programmet, hvor der er blevet anvendt de metoder og modeller fra systemanalysen og design.

Programmet blev afslutningsvis testet gennem en black box test for at sikre, at programmet virkede som tiltænkt. Ud fra denne test kan der konkluderes, at programmet virker, og at der på den vis er dannet en viden for testning til kommende systemudviklinger.

Igennem hele forløbet er der blevet skabt en bedre forståelse for, hvordan et systemudviklingsforløb forløber. Det har derfor været særdeles lærerigt, at opleve processen fra analyse til endeligt produkt, hvor der hele vejen er en rød tråd. Ligeledes er der opnået en større viden for, hvor meget et produkt kan udvikle sig fra idéfasen og hvorledes beslutninger foretages i forhold til brugeren. Således, er der blevet skabt en stor forståelse for de enkelte modeller og metoder, da disse ofte har måttet revurderes grundet ny viden senere i processen. Dette har således givet et godt indblik i, hvorledes man arbejder med systemudvikling i erhvervslivet gennem en iterativ proces.

7 Kildeliste

7.1 Bøger:

Reference: SAD1	
Titel:	Objektorienteret Analyse & Design
Forfatter:	Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen og Jan Stage
Forlag:	Marko ApS, Aalborg
Udgivelsesår:	2001
Udgave:	3. udgave
ISBN nr.:	87-7751-153-0

Reference: DIEB9	
Titel:	Interaction Design: Beyond Human-Computer Interaction
Forfatter:	Dr. Helen Sharp, Professor Yvonne Rogers og Dr Jenny Preece
Forlag:	Wiley
Udgivelsesår:	2007
Udgave:	2. Edition
ISBN nr.:	0470018666

7.2 Internet:

Navn	**Addresse**	**Dato**
Reference: SAD2		
Skyttebogen 2009 - 2010	http://www.skytten.dk/Admin/Public/DWSDownload.aspx?File=/Files/Filer/God+viden/Skyttebogen2008-2009.pdf	21-05-2010
Reference: DIEB1		
OS Platform Statistics	http://www.w3schools.com/browsers/browsers_os.asp	21-05-2010
Reference: DIEB2		
IDA	http://www.cs.aau.dk/~dubois/docs/Lektion06.pdf http://www.cs.aau.dk/~dubois/docs/Lektion06a.pdf http://www.cs.aau.dk/~dubois/docs/Lektion06b.pdf	21-05-2010
Reference: DIEB3		
Stakeholders og personas	http://www.cs.aau.dk/~dubois/docs/Lektion03.pdf	21-05-2010
Reference: DIEB4		
Mål for interaction	http://www.cs.aau.dk/~dubois/docs/Lektion01-part1.pdf	21-05-2010
Reference: DIEB5		
Interaktionsformer	http://www.cs.aau.dk/~dubois/docs/Lektion01-part2.pdf	21-05-2010
Reference: DIEB6		
Gestaltlove	http://www.cs.aau.dk/~dubois/docs/Lektion01-part3.pdf http://www.cs.aau.dk/~dubois/docs/Lektion03.pdf	21-05-2010
Reference DIEB7		

De 4 designprincipper	http://www.cs.aau.dk/~dubois/docs/Lektion01-part1.pdf	21-05-2010
Reference DIEB8		
Interaktionsrum	http://www.cs.aau.dk/~dubois/docs/Lektion03.pdf	21-05-2010
Reference: DIEB9		
Begrebsmæssig model	http://www.cs.aau.dk/~dubois/docs/Lektion01-part2.pdf	21-05-2010
Reference: OOPA1		
Testing af program	http://www.cs.aau.dk/~normark/oopa-10/html/oopa.html	21-05-2010
Reference: OOPA2		
Object-orienteret Pro-rammering og Algoritmik	http://www.cs.aau.dk/~normark/oopa-10/html/oopa.html	24-05-2010

8 Bilagsfortegnelse

Bilag 1
Tilstandsdiagrammer for klasserne – 1 side

Bilag 2
Mockups – 1 side

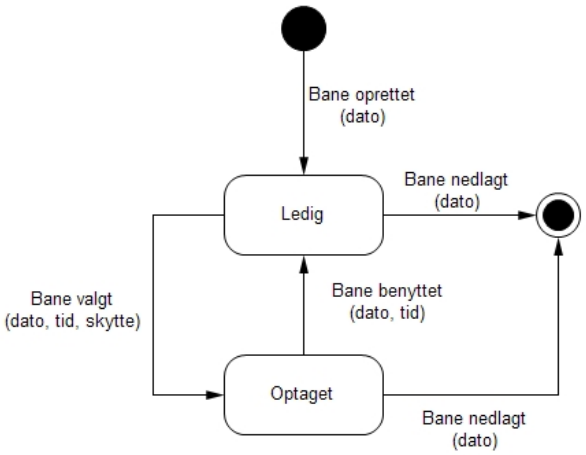
Bilag 3
Testmaterialer – 4 sider

Bilag 4
Skydeskive – 1 side

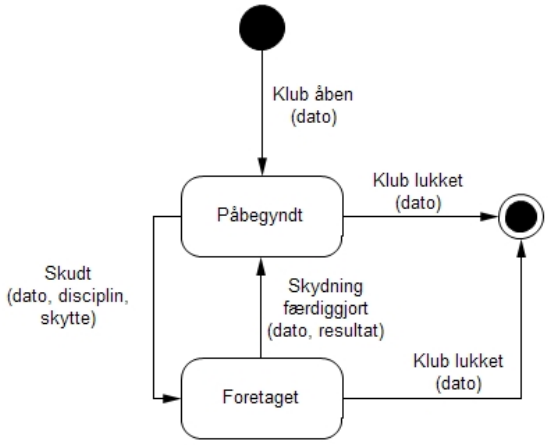
Bilag 5
Første design i Windows Forms – 1 side

Bilag 1 Tilstandsdiagrammer for klasserne

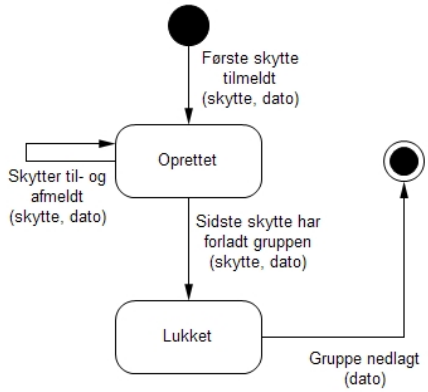
Bane



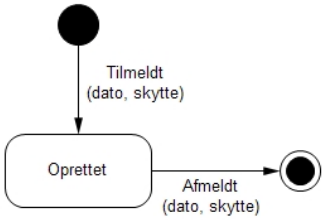
Træning



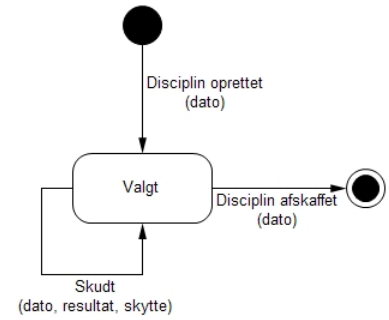
Gruppe



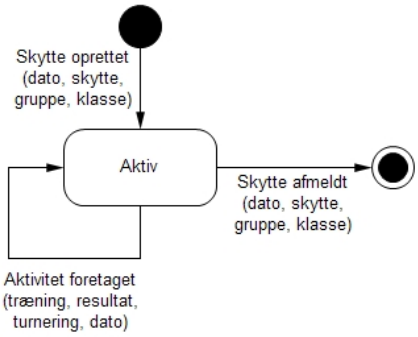
Medlemskab



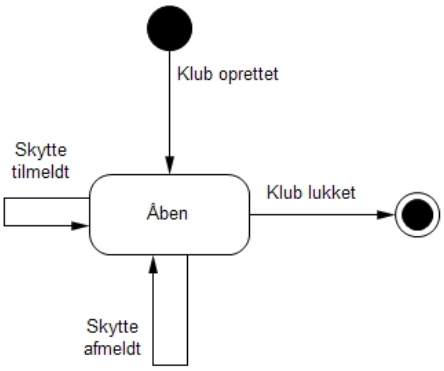
Disciplin



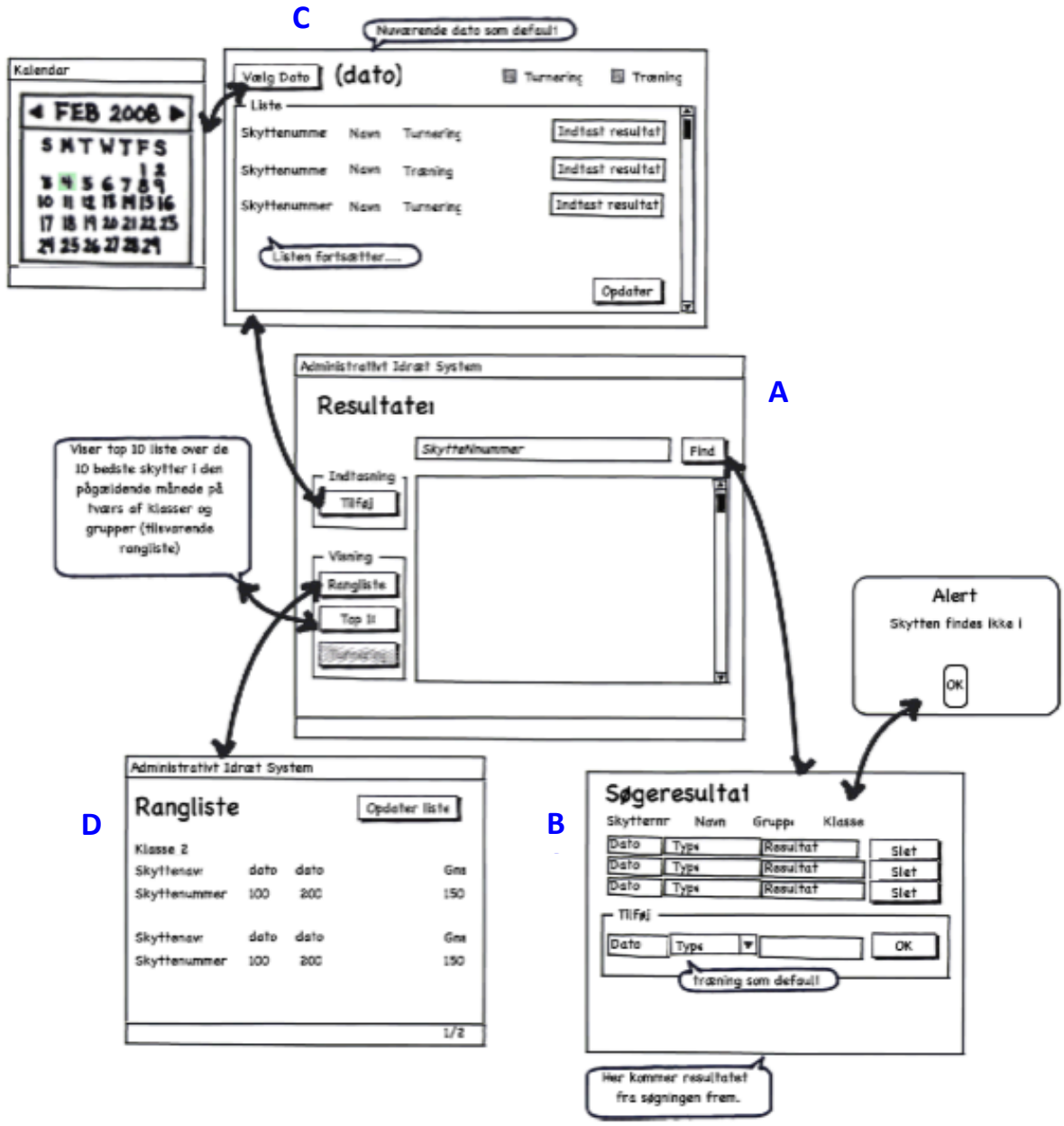
Skytte



Skytteklub



Bilag 2 Mockups



Gruppe B230, Aalborg Universitet
Vejleder Jesper Kjeldskov

Bilag 3 Testmaterialer

Testintroduktion

Indledning
Tak fordi du vil deltage i denne usability test. Jeg læser følgende højt for at sikre at alle vores testpersoner får det samme udgangspunkt, og for at være sikker på at jeg får det hele med.

Hvad skal testes?
Du skal i denne test arbejde med en demonstration af et administrativt system over skytteresultater, hvilket skal gøre det nemmere for en skytteklub/forening at holde styr på disse data.

Hvorfor denne test?
Vi vil gerne undersøge brugervenligheden af vores system, således at vi kan få en indsigt i, hvad brugeren tænker og forventer. Derfor har vi lavet nogle opgaver til dig, som du skal løse ved brug af systemet. Husk at det udelukkende er systemet vi tester og ikke dig som person.

Hvordan?
Under hele testen vil jeg sidde her som støtte, hvis du skulle gå i stå, har spørgsmål til testen eller lignende. Jeg er her ikke for at overvåge din testsession, men vil af naturlige årsager følge testen.
Du starter med at læse opgaven højt, hvorefter du fortæller mig hvordan du forstår opgaven. Dernæst fortæller du, hvorledes du ønsker at løse opgaven, hvorefter du går i gang med benyttelsen af systemet. Imens du løser opgaven vil vi gerne have at du tænker højt, hvilket betyder at du fortæller hvilke tanker du gør dig i det du færdes i systemet. Det er ikke naturligt at tænke højt, hvorpå der er sandsynlighed for at du kommer til at glemme det, hvorfor jeg vil minde dig om det hvis det sker. Hvis du går i stå under testen må du godt spørge mig, men jeg vil ikke hjælpe dig direkte, da dette vil være kompromitterende for testen. Jeg vil i stedet forsøge at hjælpe dig til selv at komme videre. Når du mener du er færdig med opgaven skal du huske at gøre mig opmærksom på det, så vi ikke er tvivl. Efter testen vil der blive foretaget en række eftersessionsspørgsmål, hvor vi ønsker besvarelser på dit overordnede af programmet.
Er du i tvivl om noget? Spørgsmål til testen, eller lignende?
Før testen starter, vil jeg bede dig underskrive en samtykkeerklæring for at sikre, at du er indforstået med rammerne for testen.

Opgaver

Forstil dig at du befinder dig i skydeklubben d. 10 februar kl. 18:30.

Opgave 1

En skytte ved navn Kristian Albech kommer hen til dig og vil gerne have indtastet et træningsresultat fra d. 9. februar kl. 18:00 ind i systemet.
Han har skyttenummeret 3917 og har opnået et resultat på 176.

- *Tilføj resultatet til skytten Kristian Albech.*

Opgave 2

Den 6. februar var der turneringsdag, Jeppestævne, og resultaterne fra skydetidspunktet kl. 12:00 er ikke skrevet ind endnu.
Følgende skytter skød på dette tidspunkt:
Navn: Morten Nielsen, Skyttenummer: 3913, Resultat: 200
Navn: Kristian Albech, Skyttenummer: 3917, Resultat: 178
Navn: Pernille Nielsen, Skyttenummer: 3915, Resultat: 170

- *Tilføj resultater til skytterne der har tilmeldt sig til denne turnering.*

Opgave 3

Du vil gerne vide hvem der ligger på 2. pladsen i top 10 listen over skytter med de bedste resultater.

- *Find frem til hvilken skytte der ligger på plads nr. 2 i top 10'en.*

Opgave 4

Du er interesseret i at vide hvilken plads en skytte ved navn Morten Nielsen har på ranglisten.

- *Find ud af, hvor Morten Nielsen befinder sig på ranglisten (gruppe, klasse og placering).*

Opgave 5

Skytten Kristian Albeck som før har fået indtastet sit resultat kommer hen til dig igen. Han fortæller at der er sket en fejl og at han har fået indtastet et forkert resultat. Det var ikke 176 men 186. Træningsdagen var d. 9. februar kl. 18:00 og han har skyttenummer 3917.

- *Slet resultatet 176 som du før har indtastet for skytten Kristian Albech og tilføj resultatet 186 i stedet.*

Opgave 6

Du er interesseret i at vide hvem der gjorde sig bedst i sin gruppen ungdom og klasse 1 på træningsdagen 2. februar kl. 19:00.

- *Hvem lægger bedst på denne dag?*

Opgave 7

Kristian Albech kommer op til dig og vil gerne have indtastet et resultat for træningsdagen d. 11. februar kl. 18:00. Han har skyttenummer 3917.

- *Tast resultatet ind for denne dato?*

Opgave 8

Skytten Nirojan Srikandarajah vil gerne kende sit gennemsnit og beder dig finde det for ham.

- *Søg på skytten Nirojan Srikandarajah, som har skyttenummeret 3914 og find hans gennemsnit.*

Eftersessionsspørgsmål

- *Var det en logisk/ulogisk opbygning og reagerede det som du forventede? (lille, mellem eller stor grad)*
- Var det nemt/tilfredsstillende for dig at: Søge, vise resultater, indtaste resultater og inddele efter træning/turnering?
- Er der nogle informationer du savner i det du har set?
- I forhold til skytteregler og inddelingen i grupper og klasser, fandt du så systemet tilfredsstillende og opfyldte det behovet?
- Stemte designet overens med hvad du forventer af et administrationssystem?
- Var det overskueligt at navigere rundt i systemet, eller savnede du noget?
- Kunne du se noget systematik/grundstruktur i opbygningen?
- Var der noget som kunne forbedres?
- Fandt du det tilfredsstillende at bruge?
- Hvordan var dit generelle indtryk af systemet?

Samtykkeerklæring

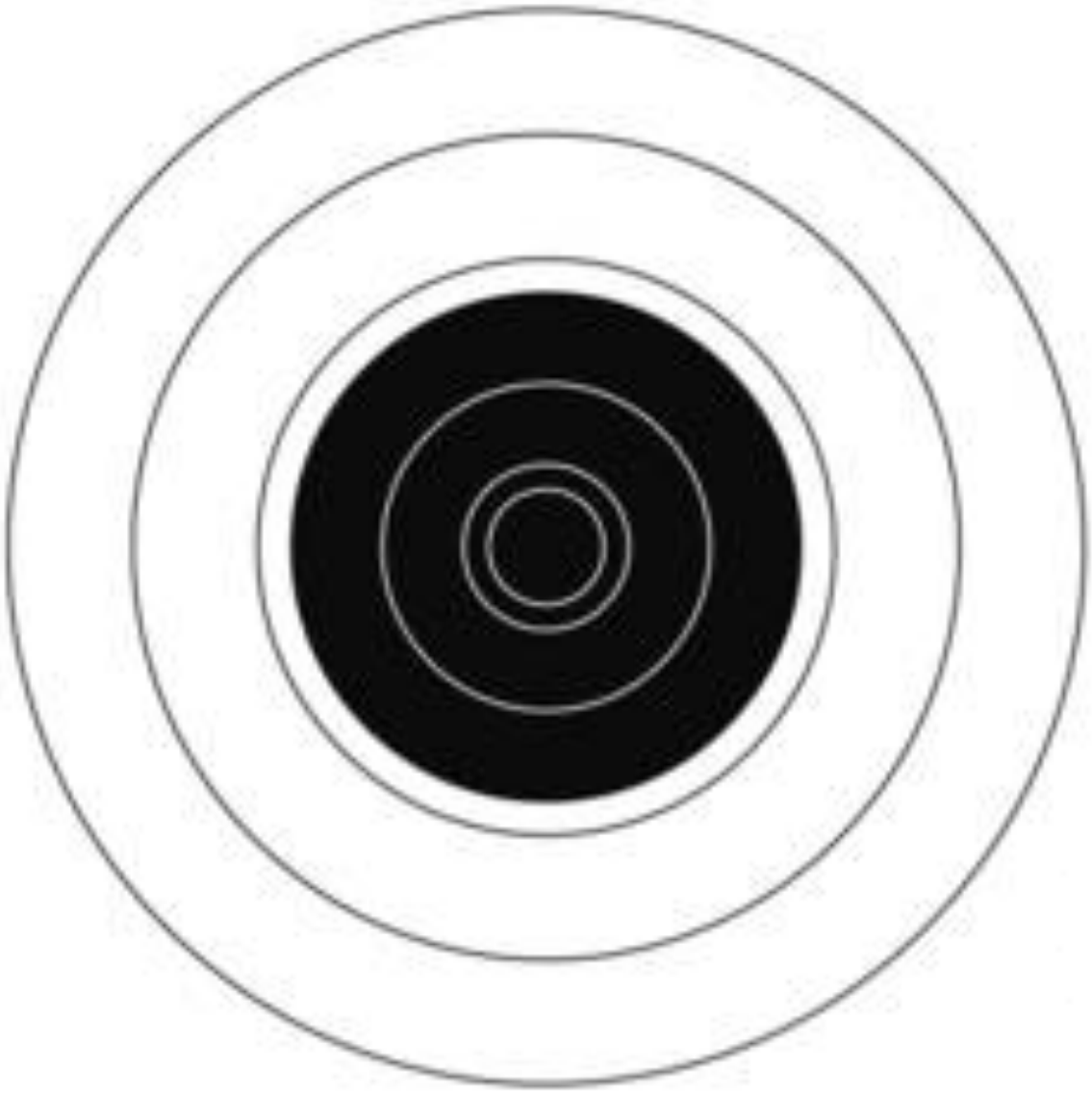
Du giver din accept på at oplysningerne fra testen må anvende af testudviklerne og vil blive brugt i den endelige rapport. Rapporten vil være tilgængelig på Aalborg Universitets database. Du vil fremgå anonym i rapporten.
Du giver hermed din tilladelse og accept til overstående gør sig gældende.
Du giver hermed din tilladelse til at vi må kontakte dig for yderlige informationer hvis dette behøves.

Dato:_____ Underskrift_____

Email:_____

Bilag 4 Skydeskive

Riffel skive 15M, cal. 5,6 mm
Skytteudstyr.com – 4119 2007
DDS godkendt 11.2007



Bilag 5 Første design i Windows Forms

Ideen med dette design var at give brugeren et overblik over resultater. Dette design skulle give brugeren en behalig oversig over de funktionelle muligheder, set ovenfra og ned. Dette design måtte ændres, da vi fandt det nødvendigt at have en fuld kalender vist, dette giver eventuelt brugeren en mere detaljeret oversigt over relevante dage og give færre taste- og klikkearbjede. Samtidigt så vi det nødvendigt at lave vinduesstrukturen om således at brugeren nu har informationsvisning adskilt fra brugerindtastninger.