IT System til administration af UEFA Champions League

P2-projekt 2009:
IT System til administration af UEFA Champions League

Vejleder:
Kurt Nørmark

Gruppe B122:
Michael K. Nielsen
Christoffer H. Poulsen
Daniel N. Hansen
Maxymilian A. Schou
Simon C. M. Sørensen

Aalborg Universitet
Det Ingeniør-, Natur- og Samfundsvidenskablige Basisår
BAIT og Informatik

Aalborg Universitet - Strandvejen 12-14 - 9000 Aalborg - Tlf. 96 35 80 80 - Telefax 98 13 63 93

Det Ingeniør-, Natur- og Samfundsvidenskablige Basisår

Projektgruppe: B122

Titel:	IT System til administration af UEFA Champions League
Tema:	Udvikling af software
Projektperiode:	P2, 2. feb. - 25. maj.
Projektgruppe:	B122
Linje:	BAIT / Informatik
Deltagere:	Michael K. Nielsen Christoffer H. Poulsen Daniel N. Hansen Maxymilian A. Schou Simon C. M. Sørensen
Vejleder:	Kurt Nørmark

Synopsis:

Dette projekt omhandler udviklingen af et administrations program til UEFA Champions League. Der er i den forbindelse blevet udarbejdet forskellige typer analyse- og designdokumenter, der indgår som bilag i denne rapport. Selve rapporten består af reflektioner over programdviklingen samt de indlæringsfaser, dokumenterne er udarbejdet under.

Oplagstal:	9
Sideantal:	88
Afleveret:	25/05/2009

Forord

Denne rapport er udarbejdet i forbindelse med et P2 projekt i perioden 2. februar til 25. maj 2009 ved den tværfaglige Bachelor i IT og Informatik linie på Aalborg Universitet under navnet "IT System til Administration af UEFA Champions League". Temaet for semestret er "Udvikling af software".

Rapportens målgruppe er vores vejledere samt andre studerende på BAIT- og Informatikfagretningen. Rapporten er skrevet i et forståeligt sprog, der gør at læseren ikke behøver den store baggrundsviden. Vi følger Harvard metoden til indsættelse af referencer, hvor kilder anføres i paranteser i den løbende tekst med forfatternavn og udgivelsesår. I litteraturlisten er der uddybende detaljer om kilderne, denne kan findes bagerst i rapporten.

Projektet består af 3 faser. En analysefase, der ligger til grund for en designfase, som så igen ligger til grund for en programmeringsfase. De to første dele består hovedsageligt af metoder og beskrivelsesteknikker. Programmeringsdelen foregår i sproget Visual C#.

Selve rapporten består af en række refleksionsafsnit, der er skrevet på baggrund af vores erfaringer i semesterets projektenhedskurser. I forbindelse med projektenhedskurserne har vi udarbejdet forskellige analysedokumenter, der er vedlagt som appendix. Der refereres til disse i refleksionsafsnittene. Afslutningsvist laver vi en samlet konklusion på baggrund af vores refleksioner.

Vi anbefaler alle der læser rapporten, at sætte sig ind i appendixet først, så de bedre kan opnå forståelse for hvilke tanker vi gør os i forbindelse med reflektionen.

Michael K. Nielsen

Christoffer H. Poulsen

Daniel N. Hansen

Maximillian A. Schou

Simon C. M. Sørensen

INDHOLD

KAPITEL 1

Indledning

Formål

I dette projekt skal der konstrueres et program, der kan administrere UEFA's Champions League (CL). Champions League består af indledende runder, derefter et gruppespil og til sidst knockout kampe og en finale. Programmet skal kunne udregne et kampprogram og stillinger ud fra de indtastede data i henhold til UEFA's regelsæt om Champions League [UEFA, 18. maj 2009b]. Programmet skal derfor kunne beregne, hvem, der går videre i de forskellige runder. Man skal kunne få en udskrift fra programmet disse skal kunne vise stillinger på forskellige tidspunkter i forløbet af CL. De specifikke krav til programmet kan ses i problembeskrivelsen nedenfor.

Problembeskrivelse

Det europæiske fodboldforbund, UEFA, arrangerer fodboldturneringer, blandt andre Champions League. UEFA har brug for et it system til at administrere Champions League. På længere sigt skal et sådant system, over internettet, give mulighed for, at mange forskellige involverede og interesserede kan indrapportere og hente data. I første omgang ønsker UEFA en prototype, der kan fungere på én PC på UEFAs hovedkontor. Data indrapporteres per brev, telefon, e-mail eller sms til en UEFA-ansat, der indtaster dem. Prototypen leverer uddata i form af skærmbilleder eller print, som den ansatte så kan sende videre i andre systemer ved hjælp af cut and paste. Prototypen skal som minimum kunne administrere et turneringsprogram, som beskrevet i reglernes artikel 6 og 7. Specielt er det vigtigt, at resultater og stillinger kan vises korrekt og overskueligt. I det omfang tiden tillader det, er UEFA også interesseret i at få understøttet et antal af de øvrige administrative opgaver.

Dog ønsker UEFA ikke, at systemet skal omfatte forhold vedrørende økonomi, reklamer, spilledragter og doping. Det er jeres opgave at analysere, designe, programmere og teste en prototype til at administrere Champions League. Den prototype, der programmeres, kan være en delmængde af det system, der er beskrevet i analysen og designet. [tnb.aau.dk, 17. maj 2009]

Formålet med projektet er at udarbejde en prototype af et Champions League administrations program. Dette skal gøres ved hjælp af en analyse, som har til formål at finde ud af, hvilke funktioner programmet skal indeholde. En programmeringsdel, hvor disse funktioner skal programmeres og en brugergrænsefladedel, som skal bygge bro mellem program og menneske. Hele programmet skal testes både via tests af grundlæggende program dele, men også af brugergrænsefladen. Brugergrænsefladen vil blive testet via en usability test, som har til formål at finde fejl og udpege eventuelle brugbarhedsproblemer.

KAPITEL 2

Refleksion

2.1 Refleksion over analysedokument

I forhold til vores analysedokument, der blev udarbejdet i forbindelse med kurset System Analyse og Design(SAD), er der sket en del ændringer i grundlaget for vores systemudvikling. Følgende afsnit omhandler de tanker, vi har gjort os omkring analysedokumentet. Der vil samtidig blive reflekteret over, hvilke dele der kunne gøres bedre. Dette afsnit også optræde, som en form for retteblad til analysedokumentet.

Problemområde og systemdefinition

Da vi startede på analysen, havde vi en klar ide om at gøre tingene så simple som muligt. Det var derfor vigtigt for os, at vi ikke fik unødige elementer med. Blandt andet fordi vores undervisning i SAD (Systemanalyse og design) havde lært os at det var en dyd, at fatte sig i korthed. Derfor mente vi ikke, at der var nogen grund til, at have flere klasser med i programmet end højst nødvendigt.
I forbindelse med udformningen af designdokument og programmet er det blevet klart, at det klassediagram, der er blevet til under analysedokumentet (se figur A.4, side 39), ikke længere er tilstrækkeligt. Det blev simpelthen alt for indviklet. Det gik ud over overskueligheden af programkoden. Det var derfor en nødvendighed, at vi måtte kigge på det oprindelige klassediagram og komme med nye ideer til klasser. Dette skulle gøres for at overskueliggøre programkoden til andre der fremtidigt skulle gennemse vores kode. I det oprindelige program var klassen Kamp den klasse, som indeholdt de vigtigste oplysninger og derfor den klasse, de fleste klasser afhang af. Dette er nu blevet revideret og klassen Hold er blevet til den grundlæggende klasse. I forlængelse af de tre klasser, der er blevet udformet under analysen (se appendix A.1 på side35), er der blevet tilføjet endnu fem klasser. En kort beskrivelse følger herefter.

Klasser

- Turnering (tilføjet):
  Turnering indeholder generelle "informationer" eksempelvis en liste af hold, der er generelt for en turnering. Turnering skal oprette en liste med hold i en turnering. Klasserne GruppeTurnering, FinaleTurnering og KnockoutTurnering arver fra Turnering, dermed er det en superklasse.

- GruppeTurnering (tilføjet):
  Denne klasse opretter lister med grupper i turneringen. Gruppens vindere samt andenpladserne går videre fra gruppespillet til KnockoutTunrneringen. Gruppespillet afgøres på baggrund af point. Modsat knockout- og finaleturneringerne, kan der ikke opstå forlænget spilletid og straffesparkskonkurrencer. Denne klasse arver en liste af hold fra Turnering.

- KnockoutTurnering (tilføjet):
  Denne klasse opretter lister med gruppevindere og andenpladser fra gruppespillet samt en kampplan for knockoutkampe.
  Et opgør skal afgøres på antal scorede mål og ikke point, som i gruppespillet. Der er to hold i et opgør og i alt 8, 4 og 2 opgør for henholdsvist 1/8, 1/4 og semifinalerne. Et hold skal spille både på hjemmebane og udebane, det vil sige der er to kampe i et opgør. Kampen afgøres på baggrund af ordinær spilletidsmål Derefter forlænget spilletidsmål og eventuelt straffesparkskonkurrencemål.
  Klassen kan arve eksempelvis hold fra Turnering.

- FinaleTurnering (tilføjet):
  Finaleklassen skal sortere to hold og give vinderen af finalekampen tilbage. Det specielle ved finaleklassen er, at der kun opstår 1 knockout kamp i modsætning til de tidligere knockoutrunder. Et opgør skal afgøres på antal scorede mål og ikke point som i gruppespillet. Der er to hold i et opgør, men kun en kamp modsat knockoutrunderne. Kampen afgøres på baggrund af ordinær spilletidsmål. Derefter forlænget spilletidsmål og eventuelt straffesparkskonkurrencemål.
  Klassen kan arve hold fra klassen Turnering.

- KnockoutKamp (tilføjet):
  Under Knockoutklassen skal der være en metode, der kan sortere to hold (en vinder og en andenplads fra gruppespillet) og give en vinder tilbage. Det specielle ved knockoutklassen er, at der i 1/8-finalerne opstår to kampe. Her skal andenpladsen fra gruppespillet skal spille første kamp på hjemmebane og vinderen fra gruppespillet skal spille første kamp på udebane. Hvem de forskellige hold skal spille mod, foregår ved lodtrækning. En kamp skal afgøres på antal scorede mål og ikke point.

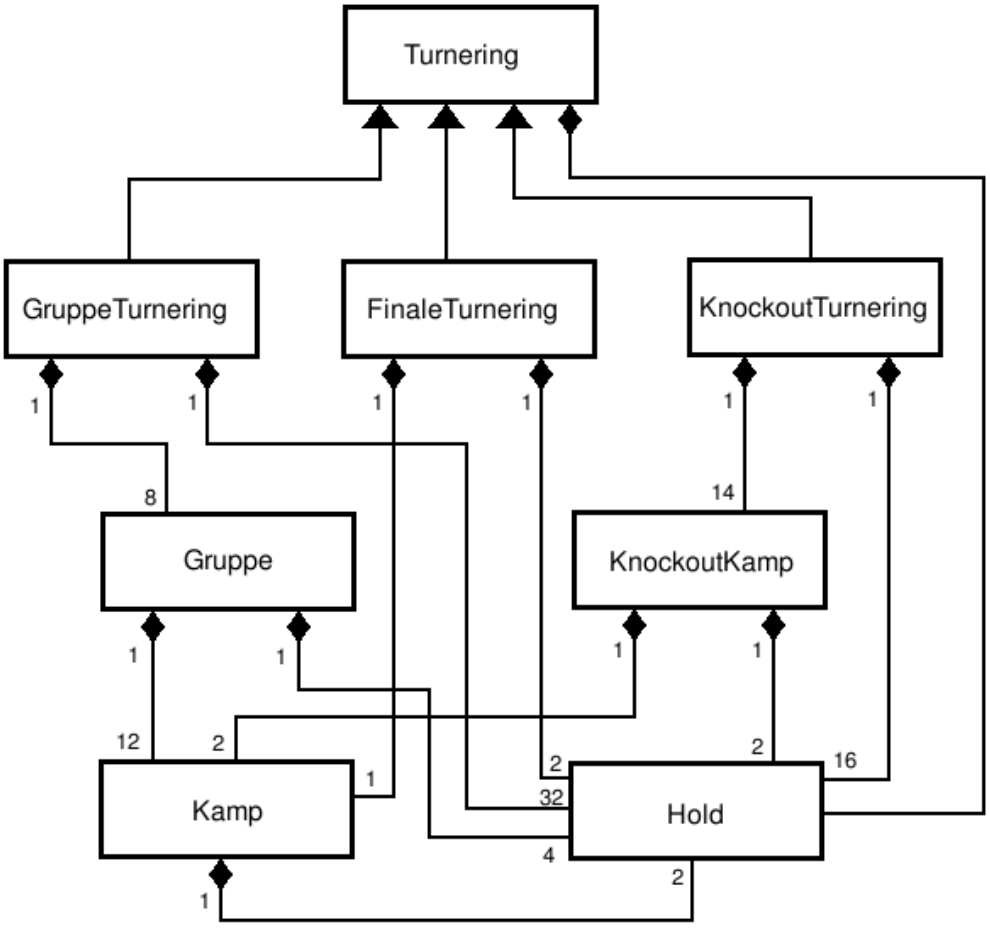
Ved uafgjort afgøres kampene ud fra udebanmål. Derefter forlænget spilletid og eventuelt straffesparkskonkurrence.

- Hold (ændret):
  I modsætning til, hvad vi skrev i analysedokumentet, har vi nu fundet ud af at mål imod og mål scoret, for holdet, kommer til at ligge under klassen Gruppe i stedet.

- Kamp (ændret):
  Klassen kamp indeholder hjemmemål og udemål, der knytter sig til et hjemmehold og et udehold. Samtidig indeholder den en metode, der kan tildele point til holdet.

- Gruppe (ændret):
  Mål imod og mål scoret kommer ind under denne klasse, idet vi mener at disse knytter sig til Gruppe som tidligere antaget. Den indeholder en liste over gruppens hold og kampprogram. Gruppespillet bliver afgjort på point. En uafgjort kamp giver 1 point, vundet 3 point og en tabt 0 point. En gruppespilsfase bliver afgjort, når alle 4 hold fra gruppen har spillet mod hinanden 2 gange.

Det blev klart, at ikke alle kampe kunne ses som gruppespilskampe og der er derfor blevet lavet en klasse for hver af stadierne i turneringen. Dette blev gjort, fordi regelsættet ændrer sig for hver af de tre stadier. Det nye klassediagram for disse kan ses på figur 2.1. Vi har oprettet tre klasser til, at håndtere hver af turneringens faser. De er specialiseringer af en overordnet turneringsklasse. De to af dem, GruppeTurnering og FinaleTurnering, bruger klassen Kamp; direkte, mens KnockoutTurnering bruger klassen KnockoutKamp(denne indeholder metoden til at finde en vinder), der så bruger Kamp. Alle klasser gør brug af klassen Hold.



Figur 2.1: Klassediagram for systemet.

Disse ændringer af klassediagrammet har betydet, at der er blevet tilføjet en del nye funktioner i forhold til funktionerne på figur A.13 på side 45 i analysedokumentet. Tabellen er blevet opdateret med funktionerne for de nye klasser og nogle af de gamle funktioner er blevet fjernet. Dette drejer sig om "Opdater gruppe(stilling, point)", som er blevet erstattet. Vi har samtidig klassificeret nogle af de nye funktioner som værende komplekse. Dette er gjort, da der er tale om beregningsfunktioner. De nye funktioner kan ses i tabel 2.2.

Funktioner	Kompleksitet	Type
Gem hold under gruppe	Medium	Opdatering
Opdater point(gruppe)	Kompleks	Beregning
Gem mål(gruppe)	Medium	Opdatering
Opdater hold rangering i gruppe (point)	Kompleks	Beregning
Opdater kampproggram(gruppe)	Kompleks	Beregning
Gem hold under knockoutfase 1/8	Medium	Opdatering
Gem 1/8 mål	Medium	Opdatering
Opdater 1/8 rangering	Kompleks	Beregning
Opdater kampproggram(1/8)	Kompleks	Beregning
Gem hold under knockoutfase	Medium	Opdatering
Gem 1/4 mål	Medium	Opdatering
Opdater 1/4 rangering	Kompleks	Beregning
Opdater kampproggram(1/4)	Kompleks	Beregning
Gem hold under semifinale og finale	Medium	Opdatering
Gem semifinale mål	Medium	Opdatering
Gem finale mål	Medium	Opdatering
Opdater kampproggram(semifinale)	Kompleks	Beregning
Opdater kampproggram(finale)	Kompleks	Beregning

Figur 2.2: Funktioner.

Anvendelsesområde

I analysedokumentets afsnit A.1.5 side 44 er der blevet identificeret to aktører i systemet. På dette tidspunkt var personen set som to, en der skulle indtaste og en til at aflæse data. Det var klart, at man ikke havde nogen reel brug for en decideret aflæser fordi det i bund og grund var en og samme person. Men også fordi det at fjerne den ene af disse aktørere ville øge overskueligheden af programmet. Da det var oprindelig tænkt at dele programmet op i en indtastnings- og en aflæsningsdel. Efter en del overvejelser er der dog ikke nogen ide i at have disse funktioner adskilt i to forskellige dele. Derfor bør der ikke være nogen adskillelse af programmet, da det reelt kun betjenes af en person. Det var samtidig et faktum, at dette ville kræve ekstra ressourcer i arbejdstid, fordi der ville

blive brugt mere tid på udformning og kodningen af brugergrænsefladen ville øges. Derfor er dette blevet ændret til at være en aktør. Denne aktør varetager derfor de sammen funktioner som indtaster og aflæser dog med få ændringer fra modellen fra figur A.9 side 43 i Appendix A.1. Her er "indtast lodtrækning" blevet fjernet, da den ikke længere vil forekomme som et brugsmønster. Personen vil ikke længere indtaste lodtrækningen, men derimod holdene. Detre er blevet erstattet af "Indtast hold". De tilbageværende brugsmønstrer ligger i to forskellige grupper, nemlig indtastning og aflæsning, men er stadig udført af den samme person som set i figur 2.3. Samtidig kunne vi revidere brugsmønsteret, så dette ville passe på en person se figur 2.4

Brugsmønstre	Indtaster/Aflæser (person)
Indtast hold	+
Indtast resultat	+
Aflæsning af runde	+
Aflæsning af stilling	+
Aflæsning af kampprogram	+
Aflæsning af resultater	+
Aflæsning af gruppe	+

Figur 2.3: Aktørtabel.

Indtastning/aflæsning af data
Når en Champions League kamp er blevet spillet, modtager UEFA resultaterne. Disse gives derefter til en medarbejder, der har til opgave at indtaste dem i vores system. Først skal han/hun vælge, hvilken turnering det drejer sig om. I tilfælde af at det er første gang systemet bruges, skal der dernæst indtastes hold for hhv. gruppespil, 1/8 finale, 1/4 finale, semifinalen og finalen. Derefter kan brugeren selv vælge, hvilken funktion han/hun vælger at tilgå. Der kan aflæses et kampprogram, som bliver opdateret løbende. Alt efter, hvornår brugeren indtaster resultater fra de pågældende kampe, kan der aflæses holdinfo med alle de relevante informationer om holdene. Brugeren kan derefter vælge at printe data.

Figur 2.4: Her ses brugsmønsteret for "indtastning af data". Bemærk ændringen i, at den person, der indtaster data nu også kan aflæse og printe dataene.

Brugergrænseflade

Da ændringer i vores aktørtabel, klassediagram, funktioner og brugsmønstrer betød, at refleksionsanalysen havde ændret sig en del fra analysedokumentet, se afsnit A.1, var det naturligt, at genoverveje brugergrænsefladen. Den simple navigation fra det gamle program er i programmeringssammenhæng blevet ændret til et lidt mere kompleks fanebladssystem, se figur A.27 side 62. Dette har medført et større overblik på samme skærm, i stedet for at skulle klikke sig mellem mange vinduer for at finde de samme informationer, kan man nøjes med et.

Selve brugergrænsefladens overskuelighed har vi gjort os en del overvejelser om. Blandt andet sprog og fodboldtermer har vi især brugt UEFA's hjemmeside [UEFA, 18. maj 2009a] til at finde. Her har vi blandt andet gjort os nogle overvejelser omkring forkortelser og opstillinger. Da vi i starten havde besluttet os for at lave programmet på engelsk, havde vi en klar ide om, at det skulle bruges af en bruger, de minimum kunne tale og læse engelsk. Efter noget tids overvejelse tog vi beslutningen om at lave sproget om til dansk. Først og fremmest fordi det at lave sproget om i brugergrænsefladen, ikke kræver nogen større arbejdsindsats og vil kunne laves i en senere revidering af programmet. Det er selvfølgelig klart, at sproget skal ændres, hvis det skal bruges af UEFA.

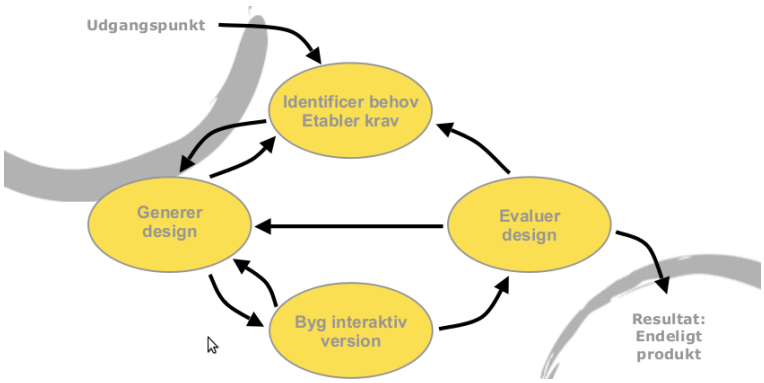
Vi valgte dog ret hurtigt, at brugergrænsefladen skulle indeholde mere konsistens. Først og fremmest fordi, vi efter OOAD kurset havde DIEB kurset, der omhandlede brugergrænseflader og tests heraf. Det betød, at vi i hele programmet ville holde nogenlunde de samme rammer og figurer(se figur A.26 side 61 til og med A.31 side A.31). Dette ville, først og fremmest, ville øge konsistensen, men også learnability, altså evnen til at lære programmets funktioner uden ad forholdsvist hurtigt. Der er ikke blevet lavet en decideret printfunktion og dette gøres stadigt ved hjælp af "Print Screen"-knappen, idet vi ikke mener, at det er en nødvendighed at have en sådan.

2.2 Refleksion over designdokument

I forhold til vores designdokument, der blev udarbejdet i forbindelse med kurset DIEB, er der sket ændringer i grundlaget for vores systemudvikling. Vores forståelse af anvendelses området har ændret sig og dette har medført ændringer i designdokumentet.

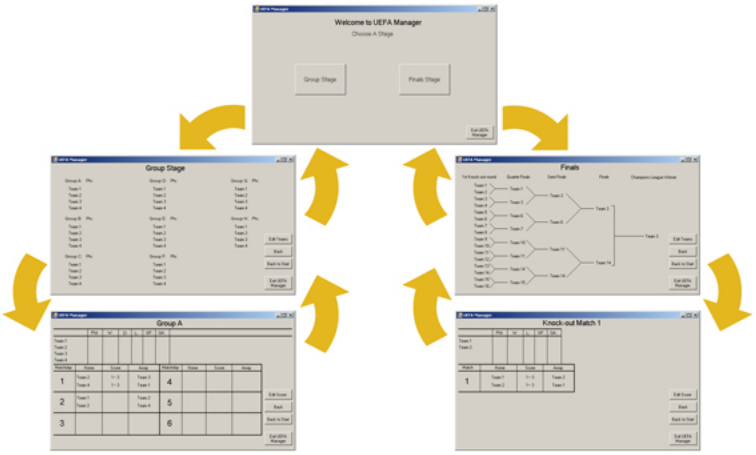
Udviklingen af et program er bygget op i fire faser: identificering af behov, generer design, byg en interaktiv version og evaluering af design(se figur 2.5). Det har også været vores intension at følge denne. Men i takt med at generering af design og bygning af den interaktive version var så tæt forbundet, blev udarbejdet af designdokumentet nærmere en slags refleksion.

Dette betød, at vi faktisk udarbejdede dokumentet i takt med implementeringen og gjorde, at de endelige ideer kom med i designdokumentet. Derfor føler vi ikke, at der burde være grund til megen refleksion over designfasen, med hensyn til modeller. Derimod var der grund til refleksion over brugergrænseflade idéen.



Figur 2.5: Model fra DIEB-undervisning: Viser de fire aktiviteter i interaktionsdesign [Skov, 18. maj 2009]

Fra starten var vi i gruppen enige om at fokuserer omkring brugsværdien af programmet. Særligt omkring, hvor effektivt programmet skulle være og hvor nemt, det skulle være at huske(2.6). Selve udviklingsforløbet har været oplæg af forskellige designløsninger. Det blev relativt tidligt i processen bestemt at vi så vidt som muligt skulle undgå popup-vinduer. Dette blev bestemt på baggrund af, hvor nemt det skulle være at huske opbygningen af programmet. Ved at holde et program begrænset til ganske få vinduer, sættes samtidig også store krav til datapræsentationen. Det samme vindue kommer til at vise mange undervinduer. Dette kan i sidste ende et give et problem, da hvert vindue kommer til at indeholde betydelige mængder information. Imidlertid viste brugerevalueringen ikke tegn på, at dette skulle være tilfældet og vi mener derfor at programmer opfylder de krav som vi har sat for det.

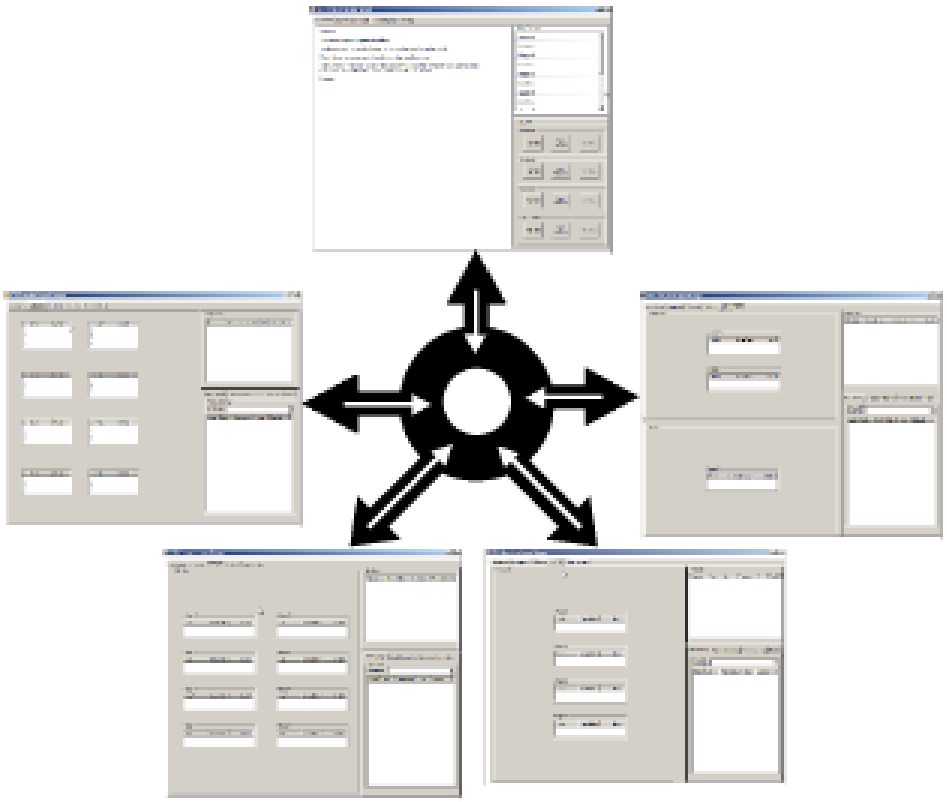


Figur 2.6: Første brugergrænseflades design ide

Nogle af de første tanker (se figur A.15 side 47) var at holde hver del af turneringen for sig selv, således hver fik sit vindue. Der ville altså således komme til at være to distinkte dele af programmet. En til indtastning af data, hvor hold og kampe kunne administreres. En anden del vil være en aflæsningsdel, hvor data kun er synlig og ikke redigerbar. Denne idé er fremkommet som indirekte resultat af vores brugsmønstermodel (Se figur 2.3 på side 16). Her ser vi nemlig 2 opgaver i programmet, nemlig at indskrive og aflæse data. Men allerede efter designet af denne brugergrænseflade virkede det besværligt at skulle linke de to dele sammen. Vi ønskede en mere fri struktur, hvor brugeres ikke skulle tvinges til et nyt vindue for at indtaste data.
Vi prøvede herefter at bevæge os tilbage til idéen omkring at holde programmet indenfor et vindue for at øge overskueligheden. Der skulle således være en slags menulinie i toppen, som kunne styre navigeringen i programmet.

Figur 2.7: Anden brugergrænseflades design ide

Dette layout (se figur 2.7) blev præsenteret for Mikael Skov under en af de sidste gange i DIEB kursus gangen. Her blev det pointeret, at den mørke farve kunne være relativt uhensigtsmæssig. Den bliver mere ulæselig og kraftig i forhold til Windows standard farve. Idéen bag den mørke farve var at følge temaerne fra de nye Windows og herved gøre programmet mere tidssvarende. I denne udgave blev alle aflæsningsmuligheder understøttet i et enkelt vindue, som man kunne navigerer igennem ved hjælp af menu linien i toppen. Dette er kendetegnet ved ganske mange Windows baserede programmer og kan derfor kun ses som en fordel for brugeren. Al indtastning skulle ske i forskellige pop-up vinduer, som vist nederst til venstre. Dette stemte ikke overens med vores idé om at programmet skulle være nemt at huske, da forskellige vinduer ville være nødvendige for at kunne styre samtlige dele af turneringen. Ud fra Mikael Skovs råd om farver og opbygning begynder vi at arbejde videre med næste løsning.



Figur 2.8: Sidste brugergrænseflades design ide

Dette layout(se figur2.8) er det sidste som vi har udarbejdet og er resultatet af en sammenlægning af de tidligere designs gode idéer. Programmet er opbygget omkring et enkelt programvindue. Al repræsentation af data er koncentreret omkring et undervindue, ligeledes med indtastning af data, dette er alt sammen vist i et fast fanebladsystem i højre hjørne. På startsiden er der placeret en hurtigstartsmenu, som guider brugeren hurtigt i gang med programmet. Bemærk konsistensen i den måde vi har lavet programmet, denne løsning har vi forsøgt, at holde igennem hele programmet.

2.3 Refleksion over programmering

I forhold til vores programmering, der blev udarbejdet i forbindelse med kurset objektorienteret programmering og algoritmik, har vi gjort os nogle tanker med hensyn til selve programmet, programtests og evalueringen. Der vil ikke optræde deciderede ændringer, men derimod nogle tanker om, hvorfor vi har gjort som vi gjorde og hvilke ting, vi kunne gøre bedre

2.3.1 Programmet

En af de største forhindringer i dette projekt har været at få programmet til at fungere efter hensigten. Vi har naturligvis været nød til at sætte nogle begrænsninger for programmet. Disse begrænsninger har fungeret både som en guideline for os selv, men også som en bestemmelse for, hvornår programmet ikke skulle videreudvikles. For der vil altid være uhensigtsmæssige ting ved et program og ting som først udvikler sig til problemer på længere sigt. Derfor har vi valgt at koncentrerer os omkring gruppespillet og ottendedelsfinalen. Kvartfinalen, Semifinalen og den reelle finale indgår i programmet, men enkelte funktionaliteter, som er understøttet i de to førstnævnte, er ikke implementeret i sidstnævnte. Dette er sket dels på grund af vores tidsfrist for programmet og dels fordi programmeringen for kvartfinalen og opefter er meget identisk til ottendedelsfinalen.

I P1 projektet blev vi præsenteret for imperativ programmering, eller måske nærmere for programmering i sin almindelighed. Dette projekt har været bygget op omkring en objektorienteret tilgang til problemstilling. Vi har således allerede før selve programmeringen fastsat alle klasser og deres funktioner (som beskrevet under analyserefleksionen). Men dette foregående arbejde, har haft ganske stor betydning for selve programmeringen. Mange af de funktionaliteter, som vi har ønsket i programmet, har kun i nogen grad været understøttet i vores klasser. Dette har resulteret i at nogle af de funktioner, som vi ønskede i vores brugergrænseflade, har krævet ekstra arbejde og implementere.

Det kunne altså have været ønskværdigt, at vi havde bestemt, hvilke funktionaliteter som programmet skulle understøtte, for herefter at få dem programmeret med fra bunden. En idé inspireret af forelæsningerne i SAD, kunne være at starte med en slags mockup. Herudfra ville det så være muligt, at bestemme layout og hvorledes data vil repræsentere sig i brugergrænsefladen. Denne løsning giver en mulighed for at bestemme et kommende programs funktioner. Dette har imidlertid været svært at realiserer på grund af semesterets opbygningen. Selve analysen munder ud i et analysedokument, som også indeholder forslag til designet af brugergrænsefladen. Men designdelen har været placeret i enden af semestret. Ligeledes var vi knapt startet på programmeringen, da analysedokumentet skulle afleveres. Vi havde således ikke nogen ide om, hvordan programmet skulle bindes sammen (med hensyn til klasser, brugergrænseflade og så videre) på daværende tidspunkt.
Først sent i processen har opnået overblik over programmets struktur, dette har også betydet at vi har måtte reviderer programmets klasser op til flere gange. Det kom der-

for som en naturlig del af projektet at revidere analysedokumentet løbende. Først under DIEB forelæsningerne fik vi et reelt overblik over, hvad der var muligt for os at lave og hvad der kunne nås tidsmæssigt.
Det har ikke været en decideret forhindring, at vi har arbejdet på så mange ting på en gang, men det har gjort det svært at danne sig et overblik over status. Man har hele tiden haft følelsen af to skridt frem og et tilbage.

Selve programmeringen har været gennemført efter princippet med bottom-up programmering. Men som tidligere nævnt, har analysen haft sine mangler og derfor har bunden ikke været solid nok fra starten. Vi har løbende måtte udvide bunden efterhånden, som funktionaliteterne blev nødvendige. Processen forløb således, at vi i begyndelsen af programmeringen "kun" havde oprettet 3 klasse. Hurtigt stod det os dog klart, at disse klasser langtfra kunne dække den problemstilling vi arbejdede med. Vi måtte altså uddybe vores forståelse af problemet, for herved at få oprettet flere klasser.

Den første brugergrænseflade, vi implementerede, var en konsolapplikation. Under udarbejdelsen af denne stod det os klart, at vi manglede en overordnet turnerings klasse, til at styre sammenhænge mellem de forskellige klasser. Denne forståelse blev dog udvidet senere, da det også viste sig nødvendigt at have en separat klasse for hver af de tre typer turnering. Vi tilføjede altså klasserne, GruppeTurnering, KnockoutTurnering og FinaleTurnering [1]. At vores superklasse Turnering [2] aldrig blev fuldt ud implementeret har været en relativ stor forhindring. Der har altså skulle oprettes flere turnerings variable. En for hver del af turneringen. Dette har resulteret i at overførslen af data objekterne imellem er blevet væsentlig mere uoverskueligt og sværere at holde fast i. Der har altså skulle større anstrengelser til at styrer objekterne og nogle af kontrolstrukturerne i brugergrænsefladen er blevet betonet af at de løser et konkret problem. Det har heller ikke været muligt at udnytte styrken bag nedarvning fra superklassen Turnering til GruppeTurnering, KnockoutTurnering og FinaleTurnering. Havde dette lykkes os, ville vi langt nemmere kunne holde snor i de enkelte hold.

Det har været nødvendigt at vende tilbage til kodningen af klasserne og at oprette nye klasser, men dette har medført at navngivningen er blevet ustruktureret. Nogle af klasserne har ikke kunne få de ønskede navne, da vi f.eks. allerede havde oprettet variable eller kontrolstrukturerer ved samme navn. Det har betydet at forskellen på mange af klasserne og variablerne, blot har været baseret på ental eller flertal endelser. Vi kunne godt have haft brugt en ensartet strategi til navngivning og fuldt den til punkt og prikke. Resultatet af dette blev at vores navngivning, både af klasser, metoder, kontrolstrukturer og variable, er blevet alt for ens. Eksempelvis kan vi nævne metoden UdskrivKampprogram. Denne metode returnerer kampprogrammet for gruppen og udskriver således intet. Her havde det været en bedre ide at kalde den noget logisk, der gav mere mening, når man skulle kalde metoderne senere i programudviklingen. Dette problem kan ses igennem

[1] I programmet kaldt GruppeTournering, KnockoutTournering og FinaleTournering
[2] I programmet kaldt Tournering

vores program, hvor navngivningen har lagt grund til diskussion.
En af grundene til at vi ikke har haft store problemer med holde styr på den lidt uheldige navngivning er, at vi valgt oprette et nyt navnerum til brugergrænsefladens klasser. Det medførte at vi eksplicit måtte fortælle, at vi nu ville oprette et objekt fra et bestemt navnerum. Havde vi lavet haft klasserne i samme navnerum eller gjort det ene projekt afhængig af det andet, ville navngivning have forvoldt flere problemer. Her har Visual Studio været os en hjælp. Vi blev i DIEB undervisningen præsenteret for en måde at oprette en Solution fil, som kunne holde styr på de forskellige navnerum. Herved har vi nemt kunne arbejde i to navnerum samtidig i Visual Studio.

Vi blev i DIEB undervisningen præsenteret for databinding i en enkelt kursusgang. Dette værktøj virkede som skabt til at løse denne problemstilling. Men at implementerer databinding i forhold til arrays virkede sværere end hvad vores evner var til. Alternativt ville vi gerne kunne have udnyttet at alle grupper i gruppeturnering er holdt fast med en liste. Denne liste er ekstremt "fleksibel" og man kunne nemt oprette grupperne ved at løbe den igennem, idet der blev trykket på en knap. Problemet med at oprette hold under listen har vi ikke kunne formå at implementere en bedre løsning for.

Vi har haft en del problemer med at oprette vilkårlige hold i en gruppe i turneringen. Brugeren skulle kunne starte med at vælge gruppe g for herefter at oprette holdene og vælge en ny gruppe uden at miste data der var indtastet. Det problem har vi kun kunne løse ved at oprette alle grupperne i gruppeturneringen med et tomt navn. Herefter har vi kunne gennemløbe listerne for hold med et tomt navn og undladt at tilføje disse hold under visningen grupperne.
Denne løsning var ikke at foretrække, idet man ikke ville komme ud for at have et hold uden navn i den virkelige verden. En anden løsning som vi blev præsenteret for senere, havde været at oprette en tom konstruktor, der kunne oprette holdende. Brugeren kunne derefter oprette holdende løbende. Denne løsning ville selvfølgelig være at foretrække i forhold til vores egen, men på grund af tidsmangel var der ikke tid til at implementere denne.

2.3.2 Test

I det følgende afsnit vil vi opridse de erfaringer, vi har gjort os med hensyn til tests samt eventuelle forbedringer.
I og med at programtestning er en proces, der har til formål at finde fejl i programmet, har vi gjort os nogle erfaringer.
Testen vi udførte var en såkaldt "Black Box" test, denne har til formål at give os viden om, om enkelte dele af programmet fungerer korrekt. Eksempelvis kunne det være om en metode giver det forventede output ved et givent input. I denne forbindelse er det vigtigt at understrege, at man ikke nødvendigvis skal teste hele programmet, derfor er forarbejdet og udvælgelsen af de klasser og man vil teste vigtig.

Det var en sådan tankegang, der gjorde, at vi besluttede at vi til at starte med ville teste de to mest simple klasser i programmet. Det drejede sig om klassen "Hold" og klassen "Kamp", primært fordi det var de to klasser alle andre klasser havde fat i. Vi fandt dog ud af, at denne beslutning var uhensigtsmæssig i og med at en god test har en høj sandsynlighed for at finde fejl.. Klassen "Kamp" havde en metode ved navn "tildel_point" hvilket var ganske fint, idet det var denne metode, der skulle tildele point i gruppespillet og her var god mulighed for at fange en fejl. Klassen "Hold" derimod, havde ikke nogen "vigtige" metoder, den indeholdt derimod nogle variabler samt properties for disse. Uheldigvis havde vi allerede lavet en test for både "Kamp" og "Hold", da vi fandt ud af at vi i princippet ikke behøvede testen for den sidstnævnte. Vi valgte i stedet og teste en anden klasse, hvilket også viste sig at være en godt ide.

Vi besluttede at i stedet for at fokusere på klassen "Hold", ville vi teste klassen "KnockoutKamp" fordi den indeholdt en metode, der kunne beregne, hvem vinderen for en knockoutkamp. Dette blev gjort via en algoritme, så dette var altså en forholdsvis vigtig metode for vores program. Dette viste sig at være en rigtig god ide da vi fandt en fejl. Metoden sorterer to hold, alt efter ordinær spilletid, overtid og straffemål og returnerer et vinderhold. Hvis stillingen forbliver lige, altså lige, vil den automatisk returnere hold 2 som vinderholdet. Dette er naturligvis ikke tilfældet, så vi ændrede i algoritmen så den ikke ville returnere et hold uden navn.

For at vende tilbage til klassen "Kamp" fandt vi ikke nogle fejl, det kan selvfølgelig altid diskuteres, hvor relevant og velbygget vores test var. Hvis der kunne opstå en fejl, vidste vi, at det måtte være i metoden "Tildel_Point". De andre bestanddele i klassen ikke var særlig relevante, ikke uden betydning, men bare så simple, at vi vidste at der ikke ville opstå fejl heri. Vi fandt ikke nogen fejl. Vi har dog set i lyset af dette gjort os nogle tanker om dette.

Da vi begyndte, at teste havde vi nogle forventninger om, at der var visse "faldgruber' i vores programmering. Vi vidste, at programmet blandt andet ikke skulle håndtere negative tal. Det vil sige, at den i stedet for at lægge målscorer til, begyndte at trække dem fra og dette var naturligvis ikke hensigten. Det var selvfølgelig heller ikke hensigten at programmet skulle kunne tage imod andre slags datatyper end det vi havde i tankerne. Dette var har selvfølgelig været i vores baghoveder under udarbejdelsen af selve testen, men set i lyset af at vi ikke fandt nogen fejl ville det måske kunne have været en ide at tage højde for dette. Da vi ikke regnede med at der ville forekomme en situation hvor man indtastede disse datatyper, havde vi heller ikke på noget tidspunkt i programmeringen tænkt på at lave nogle fejlmeddelelser.

Vi havde fra undervisningen i Objektorienteret programmering og algoritmik lært om exceptionhandeling. Denne måde at fange problemerne på kan udføres ved hjælp af en try/catch metode. Reelt siger man til compileren, at den skal afprøve koden, hvis det så viser sig, at den ikke fungerer, beder man den at afbryde og springe videre til noget andet kode. Det skal dog siges, at vi vidste, at compileren ville fange en del af de typefejl der normalt ville opstå før programmet kører. Der opstår altså en statisk typesikkerhed. Vi valgte derfor at designe vores tests ud fra en forventning om, at det ikke var fra brugerens side fejlene kunne, men derimod systemet.

Set i lyset af at vores test kun fangede en fejl(i to klasser), synes vi at testen er gået godt fordi vi fangede en fejl, man måske ikke ville have opdaget. Denne fejl der potentielt kunne have sendt et forkert hold videre, kunne potentielt have udbredt sig i programmet. Samtidig har vi lært, at hvis vi på et senere tidspunkt skulle lave tests, så ville det være en ide at tage højde for faktorerne, vi ikke regner med kan forekomme.

2.3.3 Brugergrænsefladen

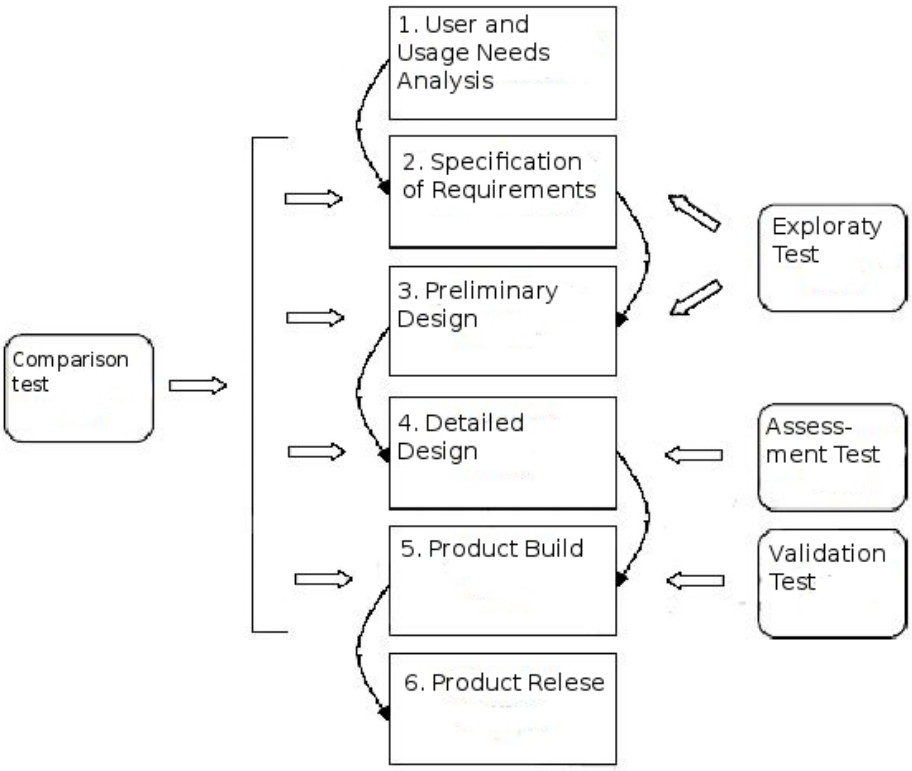
Gennem DIEB undervisningen har vi lært om forskellige modeller til afprøvning af IT-systemer samt forskellige metoder, hvorpå man kan lave opfølgende databehandling. For os har DIEB kurset været et opfølgende kursus til "Usability og Usabilityevaluering af IT-systemer (UUIT)", som blev holdt under P1 og er sammenkædet med Systemanalyse og design kurset. I UUIT blev vi bekendt med vigtigheden af evaluering under udviklingen af et system. Derfor har det også været naturligt, at evaluering af systemet skulle implementeres i udviklingsarbejdet af system. Men evaluering på forskellige stadier af udviklingen kræver en del ressourcer, så vi har hele tiden skulle overveje det eventuelle udbytte af en given evaluering.

Et af vores store problemer har været at bestemme grænser for systemet. Udviklingen af systemet har, for os, været en kreativ proces, hvor idéer hele tiden dukker op og alle har en mening om alt. Det har både fremstået som en hæmning og samtidig en fordel.

Det er interessant at personer på samme uddannelse og til en vis udstrækning samme interesser, har så forskellige meninger om hvad et godt design er. Derfor var det også vigtigt, at vi først og fremmest var enige om, hvad godt design var før vi implementerede og evaluerede det. Vi kan kun forstille os, hvor anderledes en brugers tanker er fra vores. Der er altså et paradoks mellem vores forestillinger om brugerens evner/tankegang og dennes reelle evner/tankegang. Det er netop denne forskel som brugerundersøgelserne fremstiller, det er de utilsigtede fejl, som end ikke er overvejet under udviklingen.

Allerede i "UUIT" kurset blev vi præsenteret for en model (se figur 2.9 side 2.9) for, hvordan systemevaluering kan indgå som ét vigtigt element for at forstå den kommende brugers tankegang. Der inddrages, under udviklingen, forskellige slags evaluering for at fastlægge systemets rammer. Modelen som vi er blevet præsenteret for i SAD-kurset og igen i DIEB-kurset, hvor forskellige niveauer af evaluering er blevet fortaget for bestemme systemet funktioner og funktionaliteten af denne. For vores projekt har det været væsentligt, at vi ikke har haft mulighed for at inddrage UEFA under denne proces. Vi har således ikke kunne lave en løbende brugerevaluering med forskellige brugere og prototyper. Vi har derfor ikke kunnet få brugerinput, så vi har selv skulle stå for vores input. Dette faktum kunne ses som grunden til det noget langsomme udviklingsarbejde med projektet. Alle har haft hver deres mening om en UEFA medarbejder uden nogen reelle argumenter. Vi har altså måtte fastsætte et designlayout, som har gennemgået et udviklings forløb uden løbende brugerevaluering, men det er dog til sidst at blevet evalueret af en, vi tror ligner en UEFA bruger. Dette har resulteret i en række problemer, som bygger på nogle relativ simple fejl, der bliver ganske kompliceret fordi funktionaliteten ikke understøttetes af kildekoden. Dette vil vi uddybe yderligere under refleksionerne over programmet og dets struktur.

Til opsamling af data ved usabilityevalueringen har vi benyttet os af en model præsenteret af Mikael Skov under "DIEB" kurset. Tidligere har usabilityevaluering været baseret på empiriske data opsamlet gennem transskriptioner af video materialet fra evalueringerne. Dette har vist sig ganske effektivt til at opfange problemer, men er også ganske kostbart i ressourcer/tid. Typisk regnes der med en faktor 4-6 for at transskribere et af testpersonernes evalueringer. At afsætte en sådan mængde tid til brugerevaluering, anså vi allerede i begyndelsen af projektet som et problem. Vi måtte altså finde en løsning, som angav hovedproblemerne, og som samtidig ikke krævede for mange ressourcer. En sådan metode blev vi præsenteret for i DIEB undervisningen. Michael Skov har sammen med Kjeldskov, J og Stage, J. udarbejdet en teori "Instant Data Analysis: Evaluating Usability in a Day"[Kjeldskov et al., 2004], som kan udarbejde en usabilityevaluering i løbet af en dag. Denne evaluering giver dog ikke et fuldstændigt billede af de problemer der er, men undersøgelser har vist at en langt række dem bliver opdaget, her især de kritiske. Denne metode var perfekt for os, fordi vi relativt hurtigt kunne iscenesætte et testscenario og herefter kunne lave opfølgning på evalueringen. Den opfølgende databehandling har vi herefter kunne foretage ud fra den problemliste som brugerevalueringen udmundede i.



Figur 2.9: Model for test af et systems usability.[Jensen, 14. maj 2009]

KAPITEL 3

Konklusion

Gennem dette projekt er der en række ting, som vi er blevet mere bevidste om. Dette gælder specielt forarbejdet til analyserne, da vi endnu en gang er blevet påmindet om, hvor vigtigt dette er. Det første analysedokument skulle have afspejlet systemets problem- og anvendelsesområde. Men vi måtte efter kort tid erkende, at der var nogle ting, som skulle revideres(se 2.1, side 11). Der var ting, som vi fra starten havde antaget værende tilstrækkelige for udarbejdelsen af vores projekt. Vi kan godt se, at det havde været bedre at lave et mere omfattende arbejde fra starten af. Men det har også været svært, da vi ikke har haft nogen erfaring med anvendelsesområdet. Vi er, godt tilfreds med resultatet af vores reviderede analysedokument og dette har lagt til grundlag, for en stor del af programmeringsarbejdet. Dog skal det siges at vores idé med at starte småt, kortfattet og med få klasser, ikke på nogen måde har lagt nogen hindring for vores udviklingsarbejde. Der har blot været tale om, at vi måtte dokumentere og revidere vores i forvejen eksisterende analysedokument, dette har dog også taget en del tid og arbejdsressourcer.

Med hensyn designdokumentet havde dette en nær tilknytning til vores programmering af brugergrænsefladen. Det blev faktisk udarbejdet på stort set samme tid, dette betød, at der ikke var så megen grund til, at reflektere herover(se 2.2 side 18). Resultatet af dette blev et designdokument, der lå meget tæt op af den udarbejde brugergrænseflade. Generelt kan man sige, at vores projekt har båret præg af, at vi ikke har haft kontakt til UEFA. Dette kan man også se på analysen, idet der ikke har været tilstrækkelig kontakt til anvendelses området. Dette kunne have gjort vores arbejde lettere i og med, at vi ikke behøvede at komme med gæt på, hvem brugeren var.

Det ligger ikke altid lige til at designe noget, som alle kan forstå og er enige om. Derfor var det vigtigt, at designet skulle evalueres gennem en usabilityevaluering. Også denne gjorde os opmærksomme på en række problemer, vi godt vidste eksisterede, men måske mere interessant også en lang række, vi ikke havde bemærkede. Det var vigtigt for os at evaluere programmet for at finde ud af, hvor meget disse problemer besværliggjorde brugerens udnyttelse af systemet. Evalueringen var vi godt tilfredse med, vi fandt en del problemer, hvilket var udmærket set i lyset af, at vi gjorde meget ud af udarbejdelsen af evalueringen. Vi valgte at løse de letteste af de problmer vi havde fundet og som vi kunne nå inden deadline. Det er derfor vigtigt for os og understrege, at man ved en senere revidering med fordel kunne rette de sidste af de problemer, der opstod ved usabilityevalueringen. Dertil kunne det være en fordel og lave endnu en usabilityevaluering, der ville afsløre eventuelle følgefejl af disse rettelser. Det kunne samtidig også være interessant at have testet systemet på dem, der reelt ville komme til at bruge det. Igen vil vi påpege vigtigheden af, at have haft UEFA inde over vores projekt. Ikke kun fordi det kunne vise om deres ideer lå tæt op ad det, vi reelt havde lavet, men også om programmet kunne bruges.

Det kan altid diskuteres hvor, "smukt" vores program var. Funktionaliteten af programmet er blevet, som vi ønskede. Men stadig er der en følelse af, at vi i dag kunne lave nogle dele væsentlig bedre.

De to mest iøjnefaldende problemer har været navngivning af klasser, men også af den metode vi brugte til at oprette hold (se 2.3, side 22). Problemstillingen er et resultat af, at programmeringen foregik efter den ikke reviderede udgave af analysedokumentet. Derfor er alle navnene ikke blevet gennemtænkt som en helhed, dette var vi ikke klar over på tidspunktet. Men efterfølgende står det os ganske klart, at en programmør vil være langt bedre stillet, hvis navnene var på plads fra starten. Hvis man efterfølgende skulle revidere programmet og eliminere problematikken omkring navngivning, ville det være en uoverskuelig opgave, som ville afføde en lang række problemer.

Med hensyn til den måde vi oprettede vores hold i turneringen, var vi selv meget kritiske på tidspunktet. Dette var dog mest af alt en slags løsning, som i et desperat øjeblik blev taget i brug for at se fremskridt. Problemet blev så efterfølgende at få ændret programmet til ikke at inddrage overstående løsning. Det viste sig imidlertid at være en større opgave, da vi allerede havde fået skrevet en betydelig del kode, som knyttede sig specifikt til løsningen.

Vi valgte at teste to klasser ved en såkaldt black box test. Der blev under disse test fundet en fejl i programmet. Fejlen viste sig, at være et reelt problem i hele klassen og denne måtte derfor omskrives. Dette synes vi selv var en meget positiv ting, idet vi fangede fejlene i opløbet og dermed kunne tage hånd om problemet.

Alt i alt har projektet båret meget præg af, at der ikke har været et overblik før hen mod slutningen. Når vi abstraherer fra projektet, har det været en frustrerende proces, at overskueligheden over programmet først er indtrådt så sent. Der er gået meget arbejdstid med, at skulle revidere materiale på baggrund af nyopståede konflikter mellem analysemodeller og kodning. Et eksempel kunne her være udvidelsen af klassediagrammet eller bare generel fastlæggelse af denne. Men lige så klart står det for os i slutningen af projektet, at revidering er en naturlig proces af både analyse og programmering. Det er der lige så vigtigt at påpege, at det nuværende produkt kun er fremkommet, som et resultat af revideringsprocessen. Vi ser det som en naturlig ting, at revidering af materiale der er skrevet tidligt i projektet sker i takt med, at forståelsen af problemområdet udvikles igennem projektarbejdet. Retrospektivt står det os klart, at projektarbejdet har været meget lignende den objektorienterede filosofiske tilgang til udviklingsarbejde. Dette afspejles både i projektets og programmets opbygning.

LITTERATUR

Goalpost.tv. UEFA Champions League logo, 24. maj 2009. http://www.goalpost.tv/wp-content/uploads/2008/12/champions-league-logo.gif.

J. J. Jensen. UUIT kursus slides, 14. maj 2009. http://www.cs.aau.dk/~jjj/UUIT/Lektion01+02-Janne.pdf.

J. Kjeldskov, M. B. Skov, and J. Stage. Instant data analysis: Evaluating usability in a day. *Proceedings of NordiCHI*, pages 233–240, 2004.

L. Mathiassen, A. Munk-Madsen, P. A. Nielsen, and J. Stage. *Objektorienteret Analyse og Design*. Marko Aps, 3 edition, 2001.

RenderSoft Software. CamStudio, 11. maj 2009. http://camstudio.org/.

M. B. Skov. Model taget fra DIEB undervisning, 18. maj 2009. http://www.cs.aau.dk/~dubois/Home/DIEB_files/PDF/Lektion02.pdf.

tnb.aau.dk. Projektkatalog for 2. semester BAIT/Inf, 17. maj 2009. http://tnb.aau.dk/fg/bait/intra/p2/P2_Projektkatalog_BAIT2_F2009.htm.

UEFA. UEFA's Hjemmeside, 18. maj 2009a. http://www.UEFA.com.

UEFA. Regelsæt for Champions League, 18. maj 2009b. http://www.uefa.com/multimediafiles/download/regulations/uefa/others/70/22/60/702260_download.pdf.

BILAG A

Appendix

A.1 Analysedokument

A.1.1 Indledning

I dette kapitel vil der forelægge et analysedokument. Dokumentet er blevet produceret som første led i udviklingen, af et program. Programmet er blevet efterspurgt af UEFA og skal hjælpe dem med, at administrere Champions League. Analysen er blevet til, på baggrund af undervisning samt bogen "Objektorienteret Analyse og Design"[Mathiassen, Munk-Madsen, Nielsen, and Stage, 2001].

A.1.2 Opgaven

Formål

Der skal i denne rapport udarbejdes en løsning på følgende projektoplæg fra det europæiske fodboldforbund UEFA:

Det europæiske fodboldforbund, UEFA, arranger fodboldturneringer, blandt andre Champions League. Se reglerne for turneringen 2008/2009[UEFA, 18. maj 2009b]. UEFA har brug for et it system til at administrere Champions League. På længere sigt skal et sådant system over internettet give mulighed for, at mange forskellige involverede og interesserede kan indrapportere og hente data.

I første omgang ønsker UEFA en prototype, der kan fungere på én PC på UEFAs hovedkontor. Data indrapporteres per brev, telefon, e-mail eller sms til en UEFA-ansat, der indtaster dem. Prototypen leverer uddata i form af skærmbilleder eller print, som den ansatte så kan sende videre i andre systemer ved hjælp af cut and paste.

Prototypen skal som minimum kunne administrere et turneringsprogram, som beskrevet i reglernes artikel 6 og 7. Specielt er det vigtigt, at resultater og stillinger kan

vises korrekt og overskueligt. I det omfang tiden tillader det, er UEFA også interesseret i at få understøttet et antal af de øvrige administrative opgaver. Dog ønsker UEFA ikke, at systemet skal omfatte forhold vedrørende økonomi, reklamer, spilledragter og doping.

Systemdefinition

IT-systemet skal registrere informationer og resultater af kampe. Det skal administrere kvalifikation, gruppespil og slutspil. Derudover skal det kunne bestemme kampprogrammet for de efterfølgende kampe samt holde styr på gruppespillet.
Systemet skal kunne betjenes af en medarbejder på UEFA, der samtidig ikke behøver nogen større uddannelse inden for computere. Det skal derudover være intuitivt og let at betjene. Det skal kunne køre på en almindelig kontor-PC, der har et standart Windows operativsystem.

Man kan anvende BATOFF kriterieriet til at støtte udarbejdelsen af en systemdefinition ved omhyggeligt at overveje, hvordan hvert element skal formuleres. Man kan også starte med, at beskrive systemet og derefter kontrollere om definitionen beskriver hvert af de seks elementer. Fordelen ved BATOFF kriteriet, er at man hele tiden kan holde systemdefinition op imod den, for at se om den er tilfredsstillende. Man kan eventuelt lave flere systemdefinitioner. I vores tilfælde, har vi brugt BATOFF kriteriet, så vi kan sammenligne det med vores systemdefinitioner og se, hvilken en der passer bedst.

- **Betingelser**
  Systemet skal være intuitivt, det vil sige være let at tilgå uden nogen form for uddannelse. Det skal følge gældende standarder for præsentation af data inden for fodboldterminologi. Det skal kun anvendes på 1 computer.

- **Anvendelsesmuligheder**
  Systemet skal betjenes af en medarbejder fra UEFA, der har en baggrundsviden inden for fodbold.

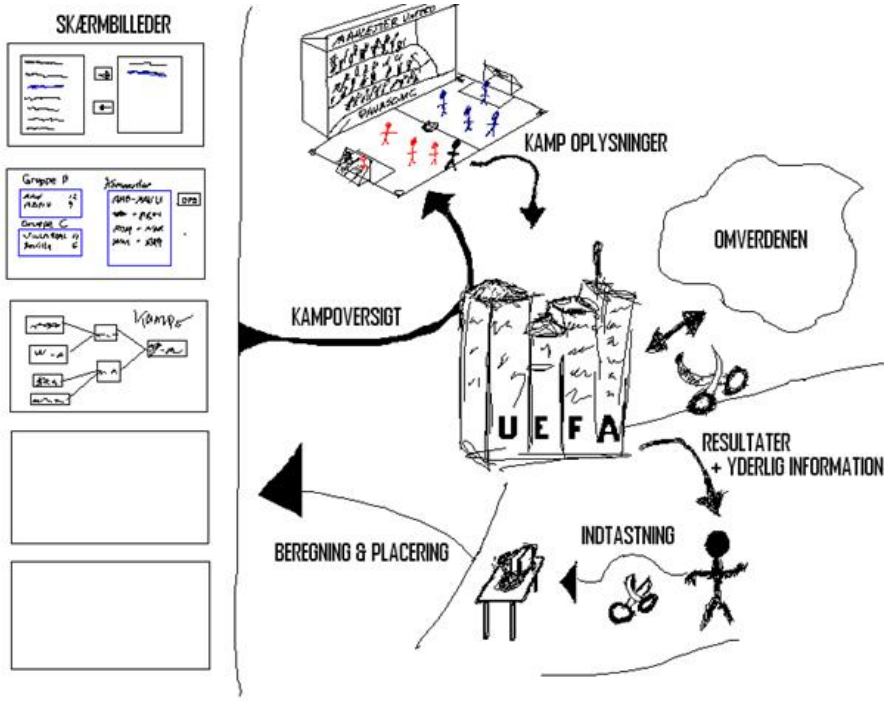
- **Teknologi**
  Systemet skal være programmeret i C#. Systemet skal kunne opereres under en standard Microsoft Windows XP SP3 opsætning.

- **Objekt**
  Gruppe
  Kamp
  Hold

- **Funktioner**
  Programmet skal være en støtte for UEFA til udformning af kampprogrammer i Champions League. Systemet skal kunne behandle input i form af registrede fodboldresultater.

- **Filosofi**
  Automatisering og generering af et kampprogram og administration af gruppespil.

A.1.3 Omgivelser



Figur A.1: Et rigt billede over systemets omgivelser.

Anvendelsesområde

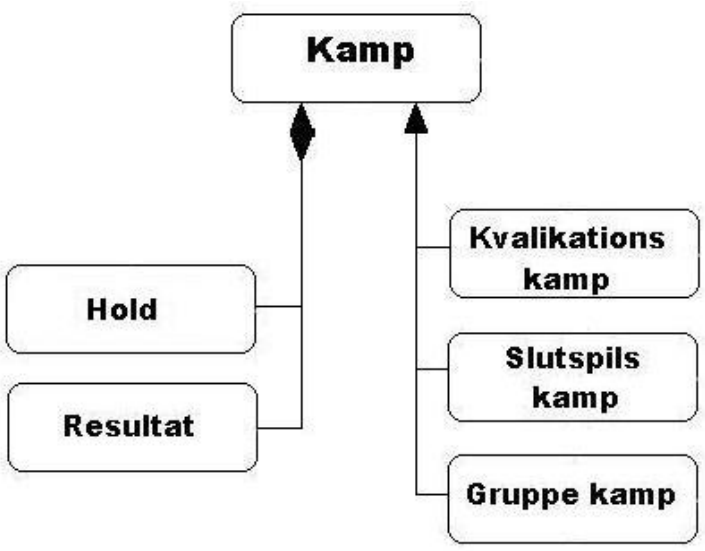
Systemet skal anvendes af en enkelt medarbejder på en standard PC. Når personen får informationer om resultater fra turneringens kampe, indtaster han dem i systemet. Den samme person skal så kunne aflæse informationer om resultater og stillinger fra turnerings cup-faser og gruppe-fasen.

Problemområde

Figur A.1 viser sammenhængen mellem systemets anvendelses- og problemområde. Systemet skal kunne administrere de indtastede data omkring turneringens kampe i de forskellige runder. Det skal være muligt for brugeren at få præsenteret de ønskede informationer omkring stillinger og resultater i de forskellige runder. Systemet skal derfor være klar over, hvilke kriterier, der afgør vindere og tabere i en given runde samt, hvilke forhold, som bestemmer stillingerne i gruppefasen.

A.1.4 Problemområdet

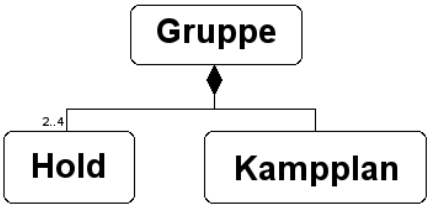
I løbet af turneringen vil klassen "Kamp" indgå på forskellige måder som set i figur A.2. I de tre kvalifikationsrunder bruges den i et cup system, hvor vinderne i hver runde findes ud fra resultaterne i deres respektive ude- og hjemmekamp. I gruppespillet deltager hvert hold så i 3 ude- og 3 hjemmekampe, der er planlagt ud fra den kampplan, som er fastlagt i regelsættet for turneringen. Når gruppespillet er gennemført bruges klassen igen i et cup system, hvor de 16 hold, der går videre til slutspillet, kommer ud i knock out runder.



Figur A.2: Struktur diagrammet for "Kamp".

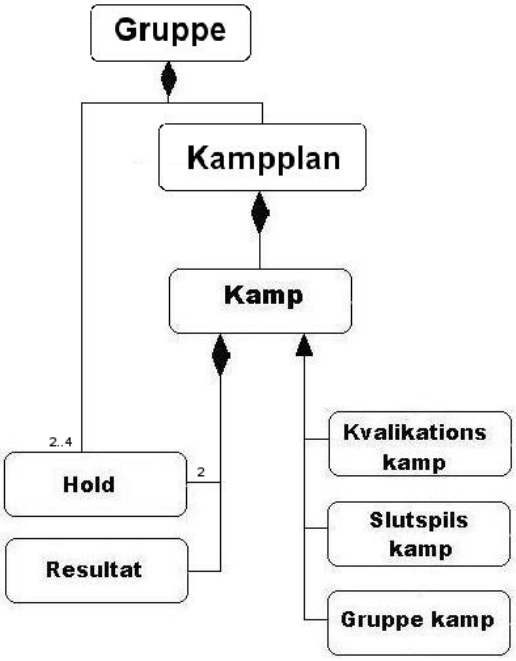
Struktur

Klassen "Hold" har ikke en omfattende generaliserings- eller aggregeringsstruktur som "Kamp", men derimod forskellige attributter.



Figur A.3: Struktur diagram for "Gruppe".

Klassen "Gruppe" aggregeres af "Hold" og "Kampplan" se figur A.3. Hver gruppe i gruppespillet får tilknyttet 4 specifikke hold, der følger den kampplan, som er fastlagt i UEFAs regelsæt for turneringen. Hver knock-out runde i kvalifikations- og finalespillet kan i princippet ses som to holds grupper, der følger de samme regler som gruppespillet, blot med andre attributter.



Figur A.4: Samlet klassediagram.

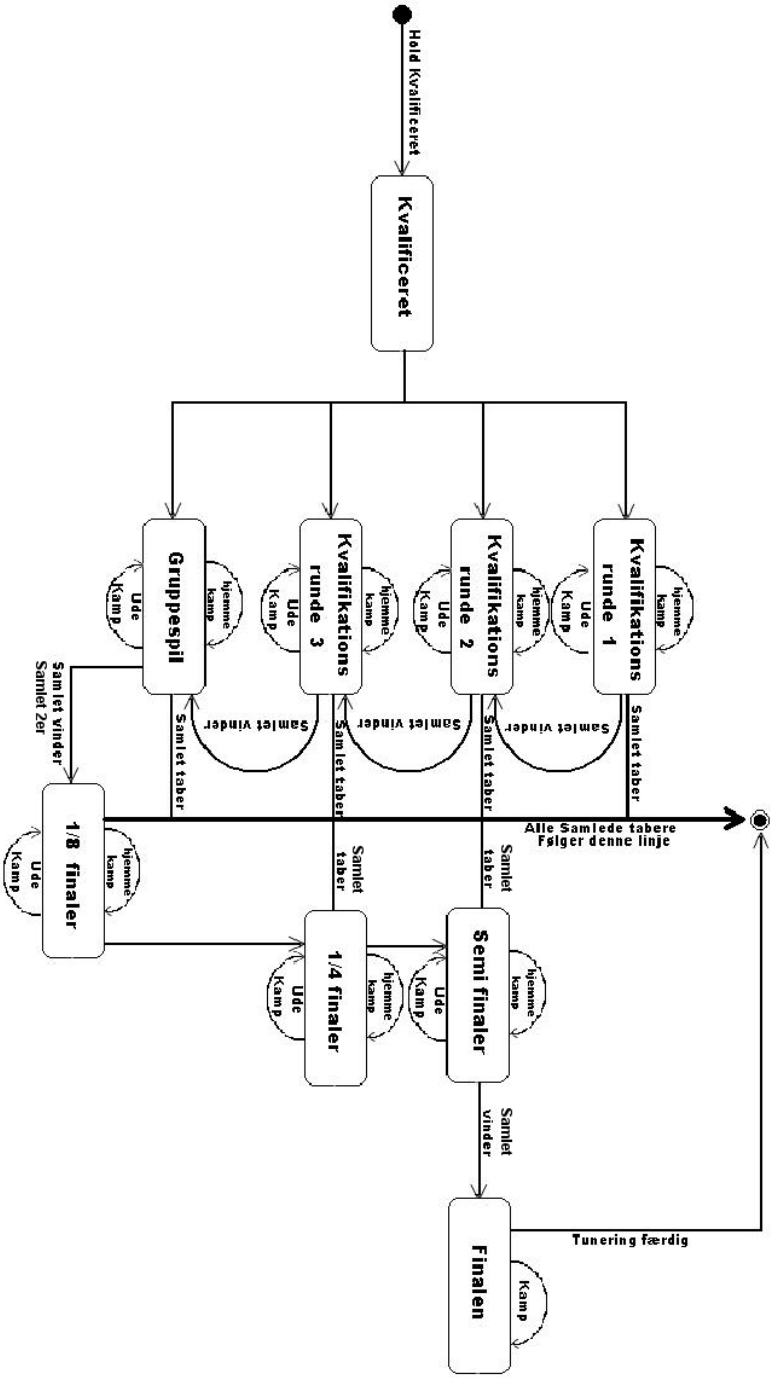
Klasser

Hvis man fokuserer på at finde de helt grundlæggende klasser ud fra de opstillede systemdefinitioner, ender man ud med at have klasserne "Hold", "Kamp" og "Gruppe".

- **Hold**
  Klassen "Hold" sammenfatter de data og attributter, der knytter sig til hvert deltagende hold. Det drejer sig om point i den pågældende gruppe, antal scorede mål og antallet af mål holdet har fået scoret imod sig og så selvfølgelig holdets navn.
  Figur A.6 på side 41 beskriver klassen Hold. Et hold skal være kvalificeret for at komme videre til gruppespillet. I gruppespillet, er der ude og hjemmekampe for holdet. Gruppespillet sender første og andenpladshold videre til 1/8 finalen, denne består igen af en ude og hjemmekamp. 1/8 finalen sender hold videre til 1/4 finalen og det samme gentager sig helt frem til finalen, hvor der kun er en kamp. I knockoutrunderne(1/8, 1/4, 1/2) og finalen kan der forekomme ordinær spilletid, overtids og straffesparksmål, som afgør kampene. Den eneste mulighed der er for at et hold ikke kommer videre er diskvalifikation.

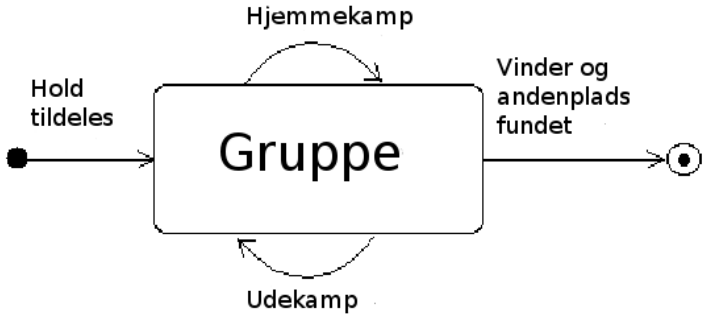
- **Kamp**
  Klassen "Kamp" beskriver de ting der knytter sig til kampe, altså hvilket hjemme- og udehold der er tale om, samt kampens resultat.
  I figur A.5 kan det ses, hvordan klassen kamp opfører sig. Den skal have tildelt et hjemme og et udehold og giver en vinder tilbage.

Figur A.5: Adfærdsmønsteret for "Kamp".



Figur A.6: Adfærdsmønster for “Hold”.

- **Gruppe**
  Klassen "Gruppe" omhandler de forhold der gør sig gældende for en gruppe i turneringens gruppefase. Dette er kampplan samt hold.
  I figur A.7 kan det ses, hvordan klassen "gruppe" opfører sig. Denne klasse skal have hold tildelt. I gruppen er der ude- og hjemmekampe, denne klasse sender en vinder og en andenplads videre.



Figur A.7: Adfærdsmønster for "Gruppe".

Hændelser

I figur A.8 vises alle de fælles hændelser for klassernes adfærdsmønstre.

	Hold	Kamp	Gruppe
Hold Kvalificeret	*		*
Hjemmekamp	*		*
Udekamp	*		*
Kvalificeret	+		
Lodtrækning	*		
Samlet vinder	*		+
Samlet 2er	*		+
Samlet Taber	+		+
Tunering slut	+		
Resultat		+	
Tildelt hjemmehold		+	*
Tildelt udehold		+	*
Turnering Færdig	*		

Figur A.8: Hændelsestabel.

I dette tilfælde var der allerede opgivet en aktør (se figur A.9), idet der i projektoplægget står, at der skulle sidde én person og indtaste og aflæse data. Vi har valgt at se denne person som to, da man kunne forestille sig, at han i princippet indtager en anden rolle når, han skal til at aflæse data, end når han indtaster. Med hensyn til de forskellige brugsmønstre forestiller vi os, at der skal være indtastning og aflæsning af data. Under indtastning skal der være hold, lodtrækning og resultat-indtastning og under aflæsning skal der være runde, stilling og kampprogram- aflæsning.
Som set i figur A.9 er der nogle områder, som overlapper både indtasteren og aflæseren. Vi forestiller os, at det er den samme person, der går ind og kontrollerer, at han har indtastet sine data korrekt, her har han bare ændret sin titel til aflæser.

Brugsmønstre	Indtaster (person)	Aflæser (person)
Indtast hold	+	
Indtast lodtrækning	+	
Indtast resultat	+	
Aflæsning af runde	+	+
Aflæsning af stilling	+	+
Aflæsning af kampprogram	+	+
Aflæsning af resultat	+	+
Aflæsning af gruppe	+	+

Figur A.9: Aktørtabel.

A.1.5 Anvendelsesområdet

Aktører

Vi har identificeret to primære aktører, som i princippet kan være den samme person. Idet personen, som indtaster data i systemet også læser dataen igennem, sikrer vedkommende sig, at han har indtastet rigtigt. Indtaster og aflæser har i alt 8 brugsmønstre (se figur A.9).

Indtaster	Aflæser
Formål: En ansat i UEFA. Indtasterens formål er at indtaste resultaterne fra de forskellige kampe.	**Formål:** Et medlem af UEFA(muligvis den samme person som indtasteren). Denne person har udelukkende til formål at aflæse data i systemet og har ingen direkte indflydelse på indtastningen.
det **Karakteristik:** Har ingen overordnet uddannelse i brug af system, men har en standard viden inden for fodbold.	**Karakteristik:** Har ingen overordnet uddannelse i brug af system, men har en standard viden inden for fodbold.
Eksempel: En ansat i UEFA, der ikke har noget større teknisk viden eller uddannelse i systemet, men som skal kunne håndtere systemet.	**Eksempel:** En ansat i UEFA(kan være den samme som indtasteren), der ikke har noget større teknisk viden eller uddannelse i systemet, men som skal kunne håndtere systemet.

Figur A.10: Beskrivelse af rollerne for systemets to aktører.

Brugsmønstre

De 8 brugsmønstre falder i to forskellige grupper:

- Indtast af hold, lodtrækning og resultat (Se figur A.11, side 45).
- Aflæsning af runde, stilling, kampprogram, resultat og gruppe (Se figur A.12, side 45).

Den første kategori har med direkte ændring af data at gøre, hvorimod den anden kategori beskæftiger sig med aflæsning.

Disse to brugsmønstre(indtastning/aflæsning) er knyttet til en eller flere aktører.

Indtast data
Når en Champions League kamp er blevet spillet, modtager UEFA resultaterne. Disse tilgår derefter til en medarbejder, der har til opgave at indtaste disse resultater i vores system. Det første, der skal gøres er, at han skal vælge om han/hun vil indtaste eller aflæse. Derefter skal han vælge den pågældende kamp og han kan så ændre i dataene.

Figur A.11: Brugsmønster for "Indtastning af data".

Aflæsning data
Når en Champions League kamp er spillet, bliver resultaterne i vores system opdateret sådan, at de kan læses udefra. Læseren kan ikke ændre i data og kan derfor kun få et overblik over kampprogram, point og resultater.

Figur A.12: Brugsmønster for "Aflæsning af data".

Funktioner

	Kompleksitet	Type
Gem hold under gruppe	Medium	Opdatering
Opdater gruppe(stilling, point)	Kompleks	Opdatering+ beregning
Opdater hold rangering i gruppe (point)	Kompleks	Opdatering+ beregning
Gem målscore (mål)	Kompleks	Opdatering+ beregning
Opdater målscore	Medium	Opdatering
Opdater kampprogram(spillede kampe)	Medium	Opdatering
Opdater pointstilling	Medium	Opdatering
Print Oversigt	Simpel	Aflæsning

Figur A.13: Funktioner.

Specifikation af funktioner

Der befinder sig 3 komplekse funktioner i systemet, se figur A.13. Disse opgaver involverer opdatering og beregning. Idet programmet selv skal kunne sortere holdene og finde ud af, hvem der går videre, involverer det, at det kan holde styr på resultater og point. Resten at opgaverne er enten af medium eller simpel sværhedsgrad. Disse funktioner beskæftiger sig med opdatering og aflæsning.

Brugergrænsefladen

Engelsk er UEFA's officielle sprog, når de skal kommunikere internt og eksternt. Derfor vil der blive brugt engelske termer igennem brugergrænsefladen af programmet. Forkortelser samt fodbold termer vil være de samme, som er brugt i UEFA's regelsæt samt på UEFA's hjemmeside (www.uefa.com).

Dialogform

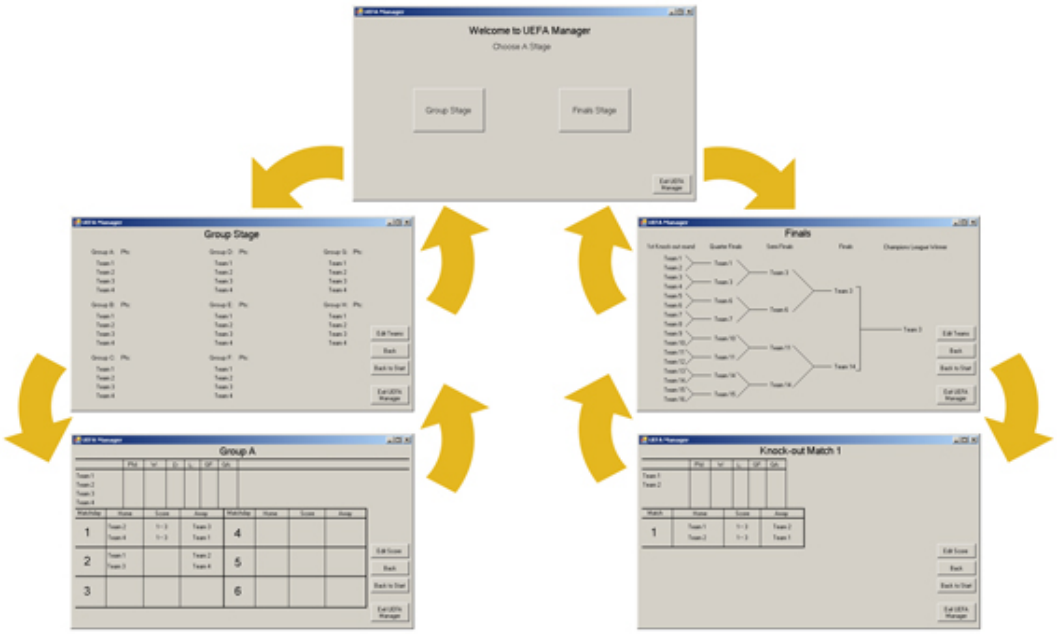
Da der er få skærme at arbejde med, er der blevet valgt en simpel navigationsstruktur der bruger knapper til at navigere med. Denne udvidelse kunne man forestille sig, at gå over til en menustruktur. Dette ville kunne give en nemmere navigation gennem større programmer.
Brugergrænsefladen har et vindue for hvert af modellens centrale klasser. Alle vinduer, på nær startskærmen, understøtter både indtastning og aflæsning. Aflæsning skal ske ved hjælp af en print-knap, som laver en kopieringsvenlig fremvisning af indtastede data. Dette vil give forskellige udskriftsmuligheder for de forskellige vinduer. A.14 viser et overblik over diverse vinduer og deres udskriftsmuligheder.

Vinduer	Udskrifter
Start	Ingen udskriftsmulighed
Gruppespil	Hele gruppespillet og point
Gruppe	gruppestilling
Finalespil	Hvilke hold der har vundet
Finalekampe	Finalekamp stilling

Figur A.14: Vinduer og udskrifter i brugergrænsefladen.

Oversigt

Figur A.15 viser et navigationsdiagram, som giver et overblik over brugergrænsefladens vinduer og relationerne mellem disse. Hver skærm bruger samme skabelon, og knapperne er bevidst placeret det samme sted for nemmere og hurtigere navigering. Disse kan dog skifte funktion alt efter, hvilken situation man er i.



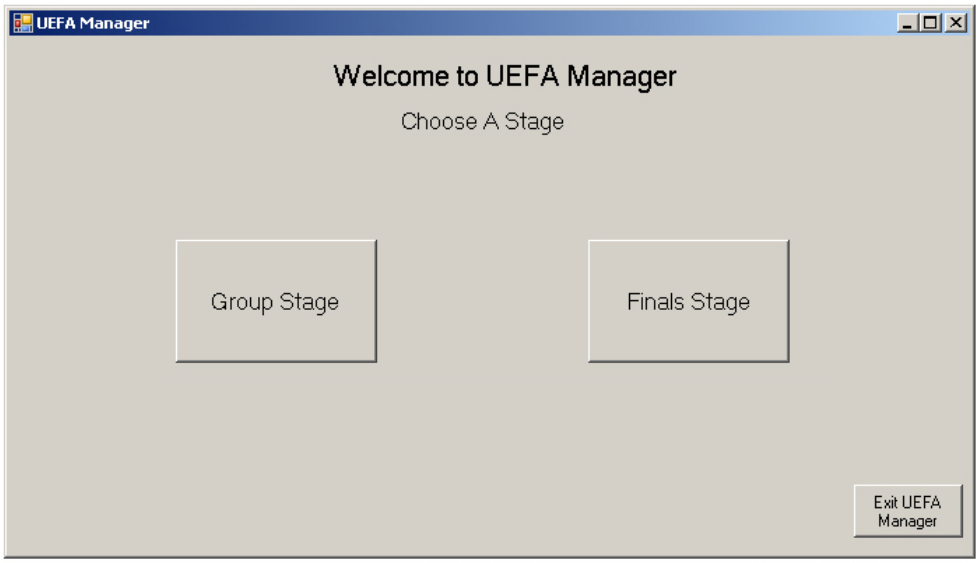
Figur A.15: Navigationsdiagram.

Eksempler

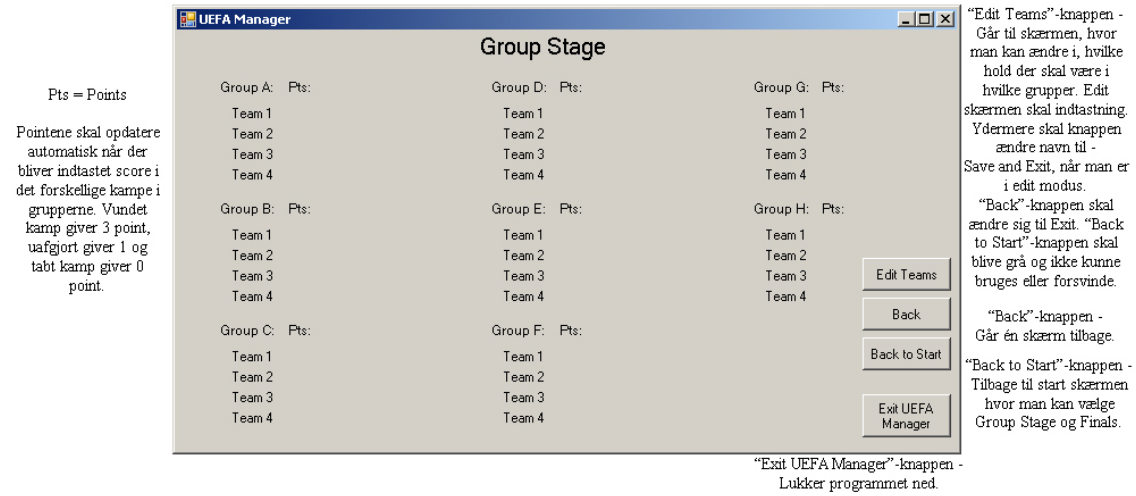
Her følger en specifikation for hvert vindue. Vinduerne er tegnet ved en blanding af Microsoft Visual C# 2008 Express Edition og et tegneprogram. Dog ville det være muligt at konstruere brugergrænseoverfladen kun i C#.
Udskrifter skal kunne vise data på en let og overskuelig måde, da dette skal kunne sendes videre. Ydermere skal der kunne copy-pastes direkte fra udskriftsskærmen over i andre medier (eventuelt som billede eller tekst).

Systemet har ialt fem vinduer:

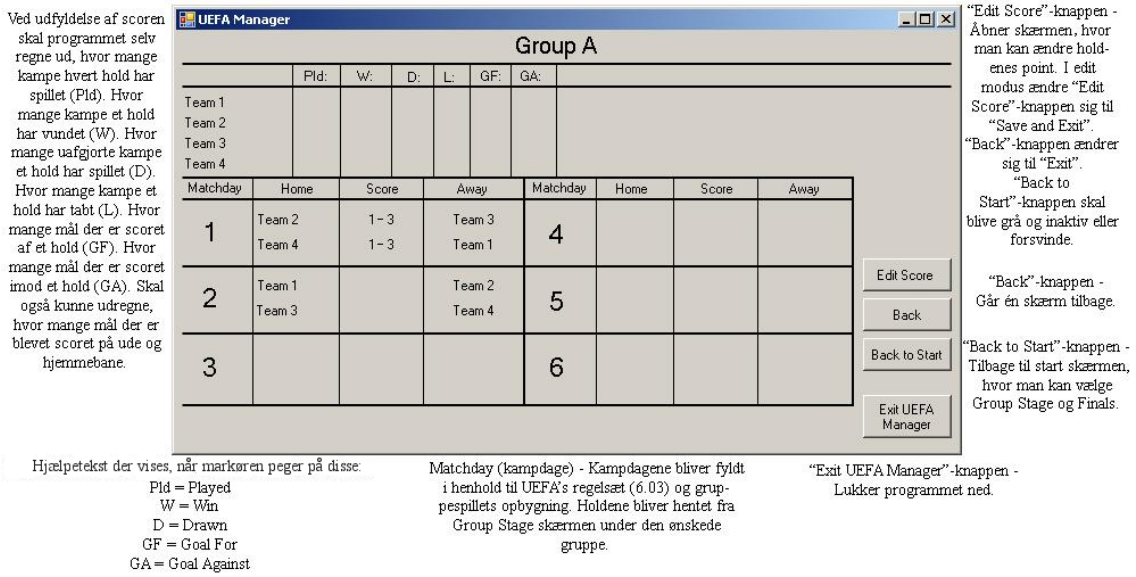
- Start skærm (se A.16, side 48)
- Gruppespilsskærm (se A.17, side 49)
- Grupperesultatsskærm (se A.18, side 49)
- Finalespilsskærm (se A.19, side 50)
- Finalekampsresultatskærm (se A.20, side 50)



Figur A.16: Startskærm



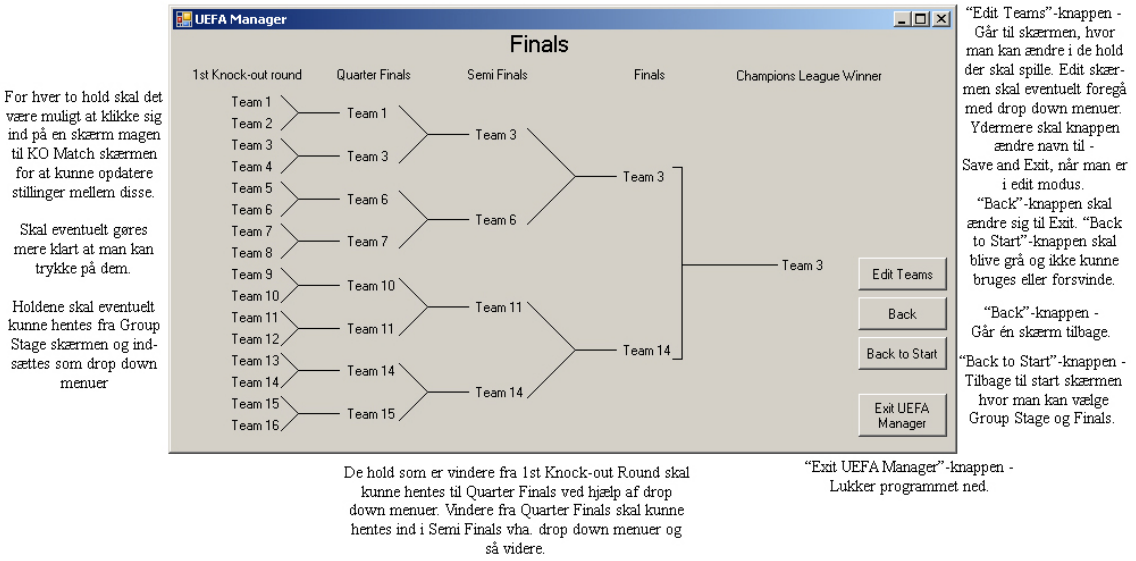
Figur A.17: Gruppespilsskærm



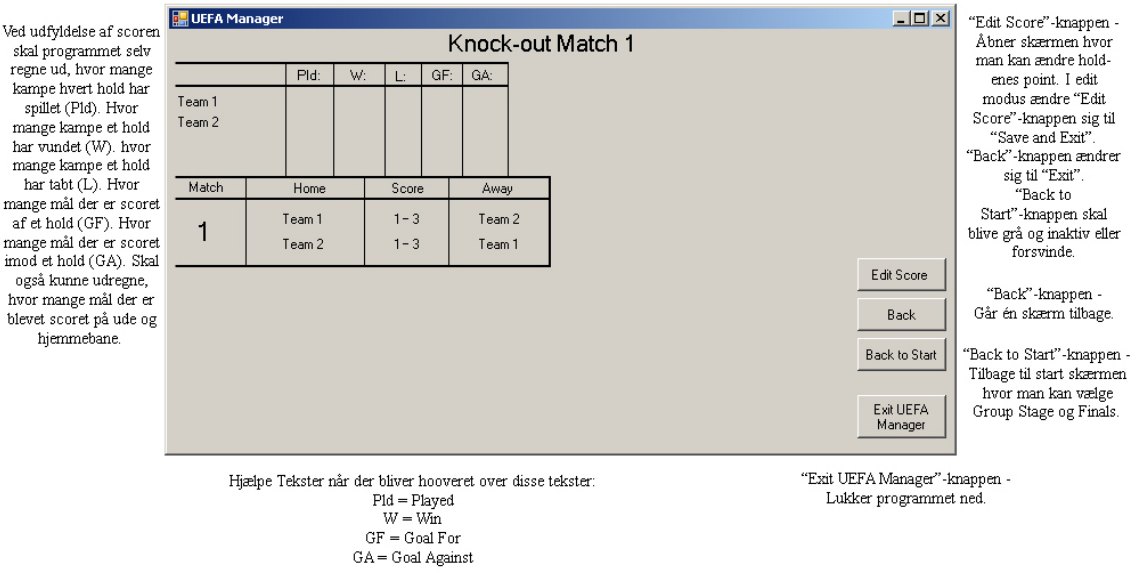
Figur A.18: Grupperesultatsskærm

Den Tekniske Platform

Systemet udvikles i C# og skal kunne afvikles på en almindelig PC med Windows XP som styresystem. Det skal anvende en vinduesbaseret brugergrænseflade og skal styres med almindelig mus og tastatur.



Figur A.19: Finalespilsskærm



Figur A.20: Finalekampsresultatskærm

A.1.6 Anbefalinger

Systemets Nytte og Realiserbarhed

Systemet har været nøje gennemdiskuteret, der har dog ikke været taget kontakt til UEFA for at lade dem komme med forslag til, hvad der skulle tilføjes. Vi har derimod brugt deres hjemmeside [UEFA, 18. maj 2009a] for at søge inspiration til, hvordan de gængse fodboldtermer og tabeller ser ud, sådan at vi kunne designe vores brugergrænseflade der ud fra. Da det er klart at det kun er en prototype, vil man stadig kunne nå at rette op på eventuelle mangler. Det skal samtidig også gøres klart, at systemet er et bud fra vores side på hvordan det skal se ud. Da vi bruger Visual C# i undervisningen, har vi valgt at basere vores system på dette. Dette betyder samtidig også, at programmet vil kunne udvides og ændres på baggrund af denne platform.

Strategi

Vi har valgt en trinvis strategi. Før vi går i gang med programmeringen er det vigtigt, at vi alle(i gruppen) er enige om, hvad vi vil og hvordan vi vil programmere. Dernæst kan programmeringen af prototypen fortsætte. Da vores projekt skal afleveres til en bestemt dato planlægger vi at stå med den færdige version af prototypen cirka to uger før aflevering.

Udviklingsøkonomi

Vi vurderer, at systemet kan realiseres på baggrund af en gruppe på 4-6 personer, der er på 2. semester i BAIT og informatik uddannelsen. De skal arbejde på projektet i 2-3 måneder, før det er klar til brug.

A.2 Designdokument

A.2.1 Opgaven

Formål

Det designede system skal hjælpe til, med at administrere UEFA Champions League. Turneringens resultater og stillinger, skal kunne indtastes og aflæses på en overskuelig måde.

Rettelser til Analysen

Vi har lavet nogle rettelser til analysedokumentet. Vi har blandt andet tilføjet nogle ekstra klasser, der kan holde styr på regelsættet for Champions league. Samtidig har vi valgt at kigge på systemet, som værende brugt af en person og ikke to, som tidligere antaget. Yderligere har vi gjort os nogle tanker om, hvad der kunne gøres anderledes. Disse kan ses i reflektionsdokumentet, der kan ses i rapporten kapitel 2, side 11.

Kvalitetsmål

Figur A.21, side 53, viser vores prioritering af designkriterier. Det er vigtigt at lave et skema, der klassificerer kvalitetsmålene for systemet, fordi man fra starten skal fastlægge, hvilke værdier ens system skal varetage. Som en tommelfingerregel vælges det at fokusere på usability, idet programmet primært vil blive brugt til administrative formål og ikke til underholdende formål.
Vi har valgt at lægge høj vægt på effectiveness og memorability, da vi mener at disse to funktioner er det man skal lægge mest vægt på, derfor er de klassificeret som meget vigtige. Efficiency, learnability, helpfulness og aestically pleasing er de næstvigtigste, derfor klassificer vi dem vigtige. Resten har vi klassificeret som mindre vigtigt eller irrelevant.

Vi vurderer at de designbetingelser vi har stillet, er til at opfylde på baggrund af en bruger med erfaringer inden for fodboldområdet og har en vis forståelse inden for computere. Systemet vil derimod ikke kunne fungere, hvis der ikke er tale om en sådan person.

		Meget vigtigt	Vigtigt	Mindre vigtigt	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	
	Aestically pleasing		X		
	Supportive of creativity				X
	Rewarding				X
	Emotionally fulfilling				X

Figur A.21: Prioritering af designkriterier.

A.2.2 Teknisk Platform

Udstyr

Systemet designes til PCer med standard Windows XP SP3, som operativ system samt .NET 3.5 framework eller højere.

Basisprogrammel

Systemet designes og programmeres i Microsoft Visual C# 2008 Express Edition, så det er muligt at anvende systemet på Windows systemer med .NET 3.5 framework installeret.

Systemgrænseflade

Indtastning og aflæsning i systemet, foregår udelukkende ved hjælp af mus, tastatur og skærm.

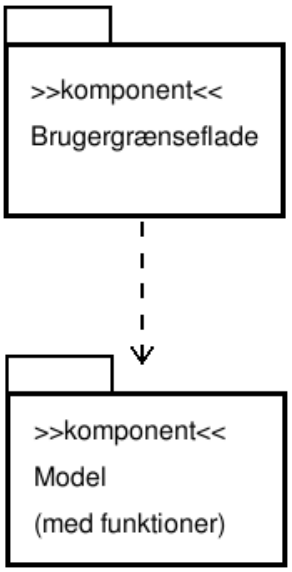
Designsprog

Designsproget er baseret på UML.

A.2.3 Arkitektur

Komponentarkitektur

Vi har valgt en todelt struktur, der indeholder en brugergrænseflade, der fungerer som grænseflade til modelkomponenten (se figur A.22).



Figur A.22: Klassediagram for systemets modelkomponenter.

Procesarkitektur

Systemet skal anvendes på en enkelt PC med en enkelt bruger. Systemet skal kunne fungere samtidig med, at andre processer i brugergrænsefladen finder sted.

Standarder

Designet af systemets brugergrænseflade, følger almindelige Windows standarder.

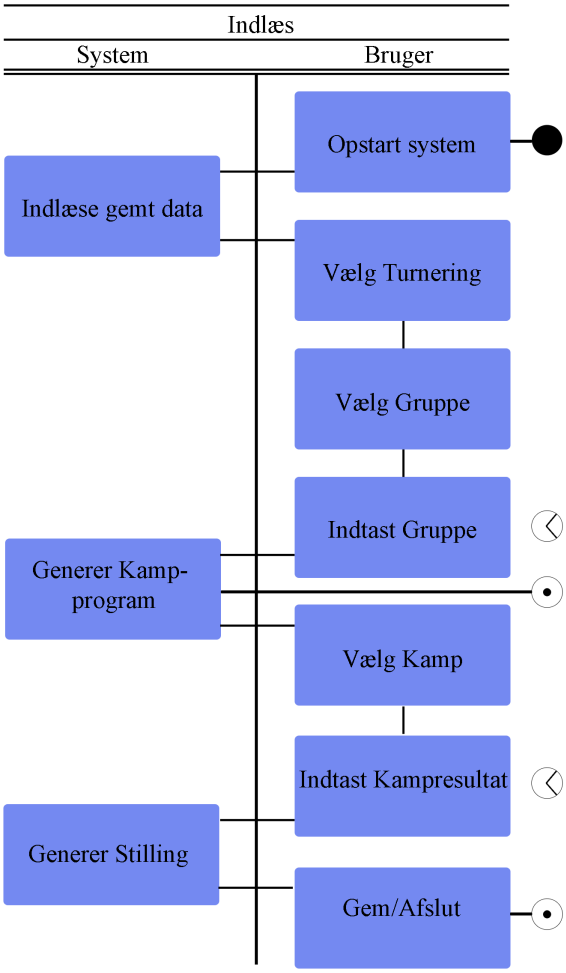
A.2.4 Modelkomponent

Struktur

I forhold til analysedokumentets klassediagram, har vi lavet nogle ændringer, der betyder at det bliver udvidet. Dette kan ses i refleksionen over analysedokumentet A.1.

Interaktion

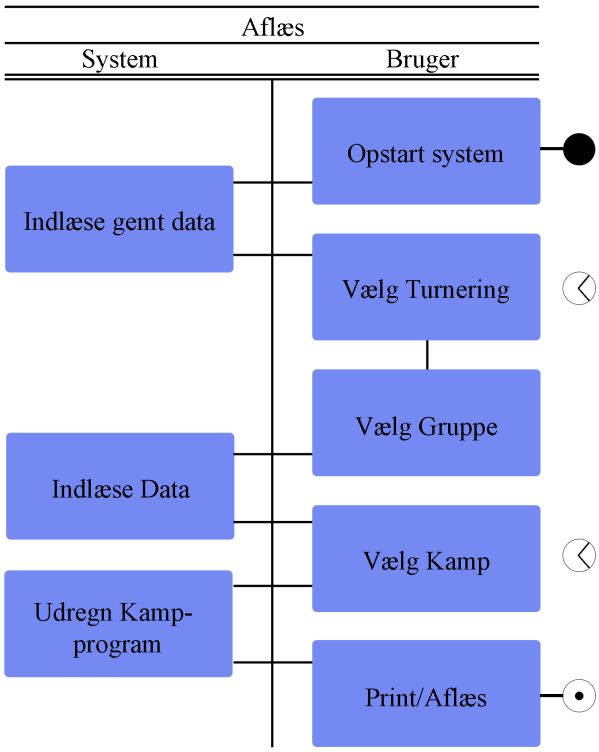
Idet vores system bygger på administrering, vil der også være store krav til brugeren, i form af indtastning. Champions League er speciel i forhold til, at knockoutfasen bliver indledt med en lodtrækning og man derfor ikke kan beregne sig frem til, hvordan holdene skal overføres fra gruppespillet. Kampresultater er også en af de ting der skal indtastes, det betyde som før, også øget interaktion med brugeren.



Figur A.23: Interaktionsmodel for "Indtast data". "Pacmans" signalerer interaktion med brugeren.

I modellen er det vist hvordan systemet interagerer, med brugeren figur A.23 under indlæsning. Ved opstart indlæser det forrige data, hvorefter systemet spørger brugeren om hvilken turnering han/hun ønsker at se en oversigt over. Som eksempel kan brugeren

vælge gruppe. Hvis det er tilfældet at der ingen hold er indtastet, skal brugeren også indtaste dette, hvorefter systemet kan generere et kampprogram.
Brugeren vælger en pågældende kamp og indtaster herefter resultater, der får systemet til at udregne pointstillingen. Derefter kan brugeren gemme og afslutte. Vi referer til figur A.13 i Appendix A.1 (side 45) for yderligere informationer om funktionerne i systemet, da ikke alle funktioner er listet her.



Figur A.24: Interaktionsdiagram for "Aflæs data".

I modellen (figur A.24) er det vist hvordan systemet interagerer med brugeren under aflæsning. Igen indlæser programmet data ved opstart. Det brugeren skal forholde sig til er, at vælge turnering, gruppe og kampe og om der skal printes.

Klasser

Herunder opremses klasserne hver især med formål, atributter og operationer.

Klasse	**Funktion**
Hold	Denne klasse laver to indkapslede variabler: string navn; int spilletkampe; Disse bliver initialiseret i konstruktoren for Kamp. Derudover betænkes det, at den skal indeholde 2 get/set funktioner, så hver af de tre variabler kan ændres og udskrives.
Kamp	Denne klasse indeholder 4 variabler. int hjemmemaal, udemaal; Hold hjemme, ude; Disse bliver initialiseret i konstruktoren for Hold. Derudover betænkes det, at klassen skal indeholde 4 get/set funktioner, samt en metode der kan tildele point til et ude og et hjemmehold, på baggrund af en pointstilling.
Gruppe	Denne klasse laver en indkapslet variabel, en indkapslet liste af hold og en liste af kampe. int point; Derudover opretter klassen en liste, der tager 4 hold(en gruppe), samt et kampprogram af en liste af kampe. Det betænkes af klassen, at den skal indeholde nogle get/set funktioner, der kan tilgå grupperne og kampprogrammet.
Turnering	Denne klasse er tænkt som en superklasse, som klasserne Gruppeturnering, Knockoutkamp og FinaleTurnering skal arve fra. Den opretter en liste af hold samt en turnering. Derudover en get/set funktion, der kan udskrive holdene.
Gruppeturnering	Denne klasse arver fra Turnering. Det betænkes, at denne klasse skal kunne oprette tre lister. To der holder styr på vindere og andenpladser af gruppespillet, samt en der indeholder grupperne i gruppespillet.

Knockoutkamp	Denne klasse arver fra superklassen Turnering. Klassen opretter en knockourturnering, der tager en vinder og en andenplads. Den opretter også to kampe, hvori der defineres hvilken rækkefølge holdende skal spille(hhv. Ude og hjemme). Klassen indeholder variable, der kan holde styr på mål for vinder og andenplads, samt straffe for vinder og andenplads. Det betænkes at denne metode skal kunne returnere, enten vinder eller andenplads alt efter målscorer, eller antal straffespark opnået.
Finaleturnering	Denne klasse ligner på en måde KnockKamp. Denne arver også fra Turnering. Klassen opretter en finalekamp(tager to hold). Ligesom unde Knockoutkamp oprettes der fire variable, der holder styr på mål og straffespark. Denne metode returnerer et af holdene, ud fra mål der er opnået.

Operation	Tildel point
Kategori	Passiv Opdatering Aflæsning Beregning
Formål	At tildele holdene point i gruppespillet, alt efter om de har vundet, tabt eller spillet uafgjort.
Inddata	Skal have to hold ind, samt et resultat.
Betingelser	
Effekt	
Algoritme	De to resultater sammenlignes, operationen ved, at resultatet er tilknyttet to hold på baggrund af hvordan de er indtastet.
Eksempel	Hjemmehold - Udehold = 3 - 2 Operationen ved nu at hjemmeholdet har 3 mål og udeholdet har 2 mål. Først ser operationen på om hjemmemål er større end udemål, i det tilfælde bliver der tildelt +3 point til hjemmeholdet, ellers ser den på om udemål er større end hjemmemål. Hvis dette er tilfældet får udeholdet +3 point. Hvis begge hold har lige mange mål, får begge hold +1 point. Operationen kaster pointene videre til propertien for Point, der opdaterer Point for pågældende hold.
Datastruktur	
Placering	Klassen Hold
Involverede objekter	hjemmehold, udehold, udemål, hjemmemål, point.
Udløsende hændelser	Når brugeren indtaster et resultat for en kamp.

Figur A.25: Operationsspecifikation for "Tildel point".

A.2.5 Brugergrænsefladen

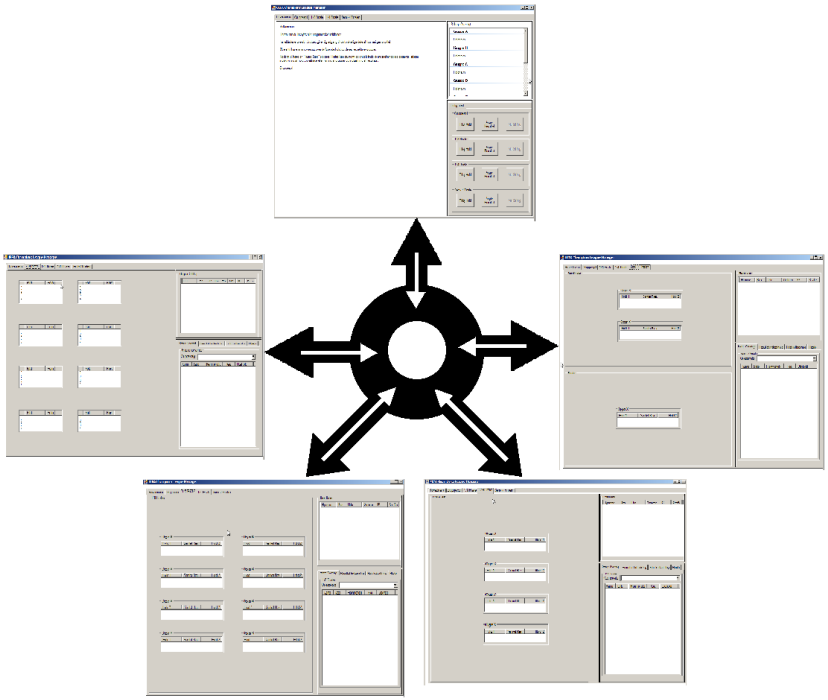
Selvom Engelsk er UEFA's officielle sprog, er brugergrænsefladen skrevet på Dansk. Dette er blevet valgt fordi programmet er produceret i Danmark og fordi der skal laves brugbarhedstestning af systemet af danskere. Eventuelle forkortelser, samt fodbold termer, vil bliver forklaret i en hjælp funktion i programmet, eller via tooltips hvis dette kan lade sig gøre.

A.2.6 Dialogform

Der er lavet en forholdsvis simpel navigationsstruktur, denne består af et enkelt skærmbillede med faneblade. Via disse faneblade, kan man komme til alle dele af turneringen. Skærmbilledet er delt op i tre felter, et stort felt til venstre, og to mindre felter til højre. Det venstre felt er et overbliksfelt, her kan man få et overblik over turneringer (f.eks. gruppespil, 1/8 Finale etc.). Feltet øverst til højre viser specificerede resultater af kampe, målscore etc. Feltet nederst til højre består ef mellem 3-4 faneblade. Her foregår indtastning af hold, til for eksempel grupper i gruppespillet, indtastning af resultater, en oversigt over kampe og en hjælpe skærm. Aflæsning skal ske via printscreen funktionen, dette vil i en senere version af programmet blive erstattet af en decideret print-knap.

A.2.7 Oversigt

Figur A.26 viser et navigationsdiagram, som giver et overblik over brugergrænsefladens vinduer og relationerne mellem disse. Hver skærm bruger samme skabelon og diverse kontrolstrukture og funktioner er placeret de samme steder gennem programmet, for at give en hurtigere og nemmere navigering. Ydermere er der på startskærmen lavet en hurtig navigerings menu nederst til højre, med en række knapper til hurtigt at sende brugeren videre. Disse skal om muligt laves til genvejstaster også, for hurtigere navigereing for den mere erfarne bruger.

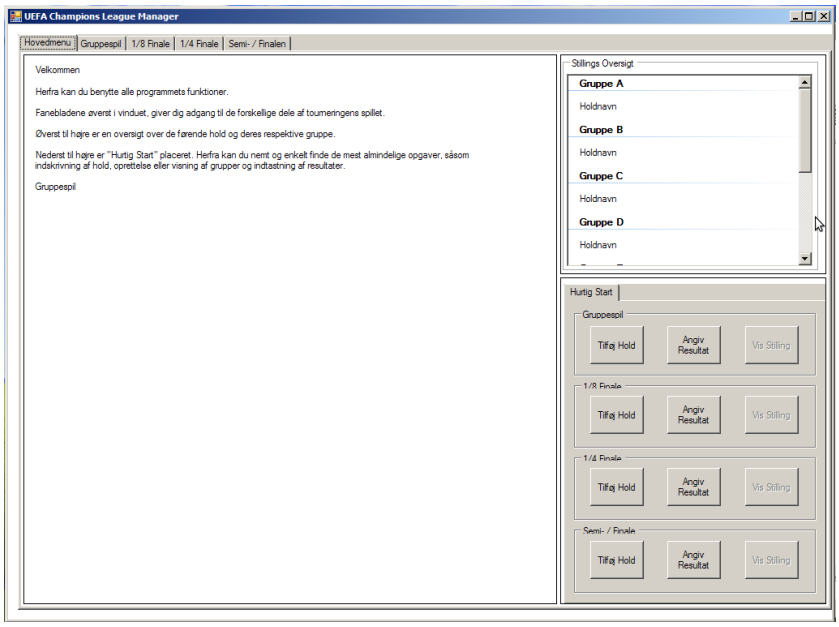


Figur A.26: Navigationsdiagram.

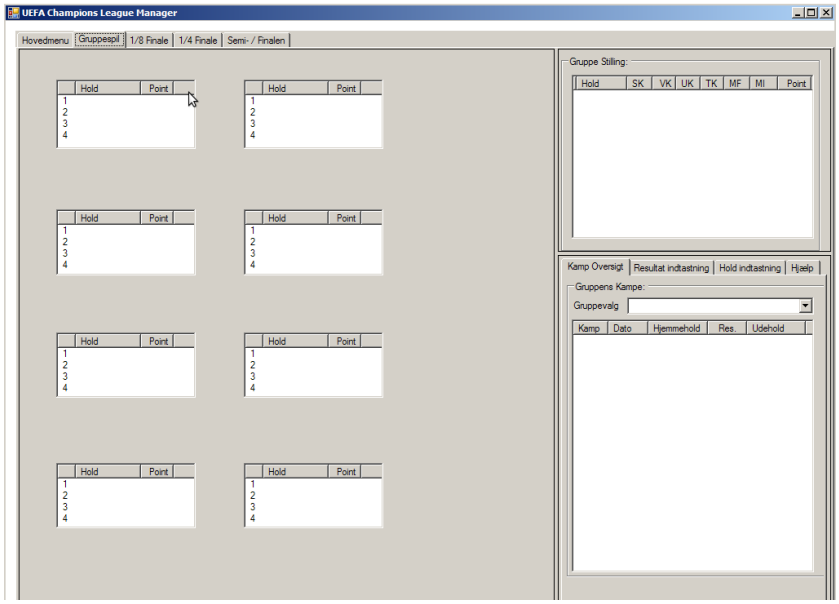
A.2.8 Eksempler

- Startskærm og det første faneblad (Se figur A.27, side 62)
- Gruppespils-faneblad (Se figur A.28, side 62)
- 1/8 Finale-faneblad (Se figur A.29, side 63)
- 1/4 Finale-faneblad (Se figur A.30, side 63)
- Semi- / Finale-faneblad (Se figur A.31, side 64)
- Feltet nederst til højre med faneblade (Se figur A.32, side 64)

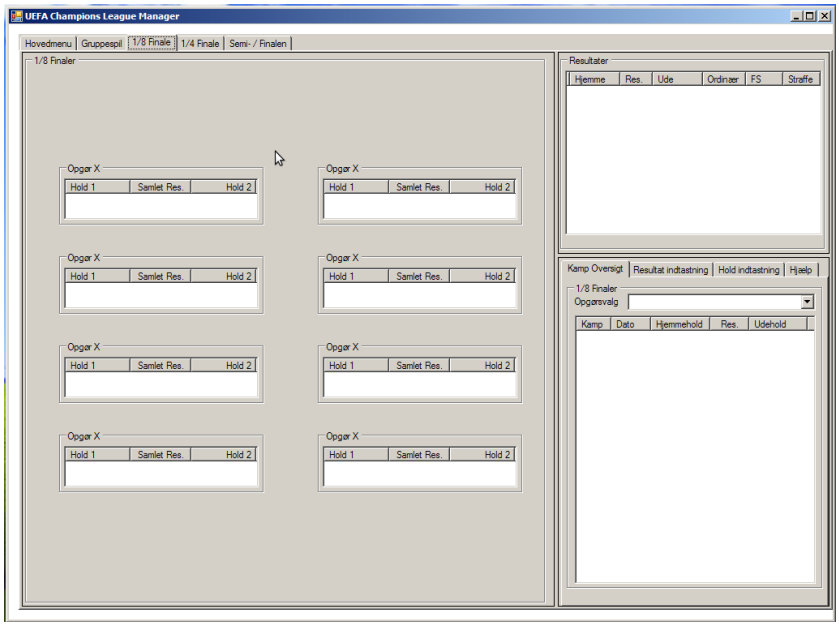
Herunder følger en specifikation for hvert vindue. Vinduerne er alle lavet i Microsoft Visual C# Express Edition (C#) og disse bør indfinde sig i det færdige program. Udskrifter vil i første omgang blive lavet med printscreen funktionen i Windows XP. Senere version af prototypen, vil dog indeholde en rigtig printfunktion, enten til papir eller til elektronisk form.



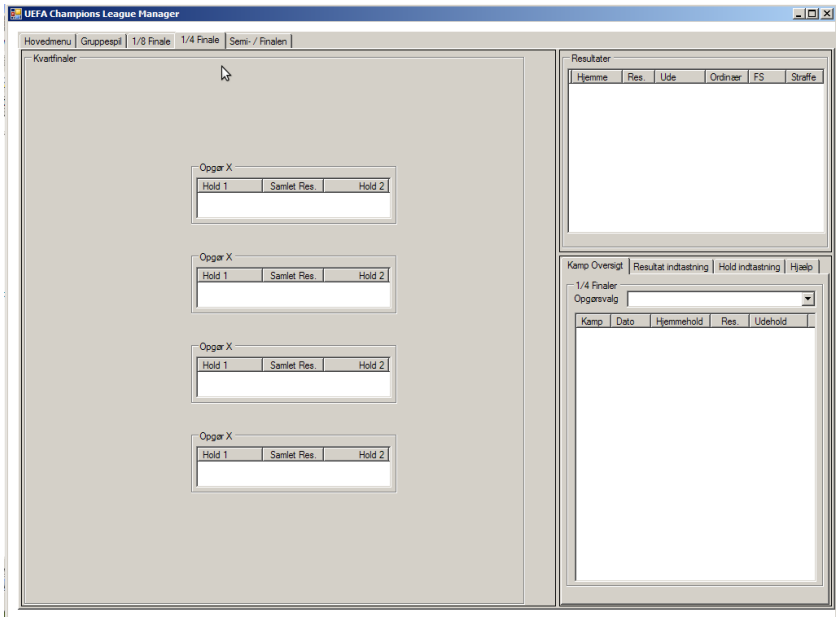
Figur A.27: Startskærm.



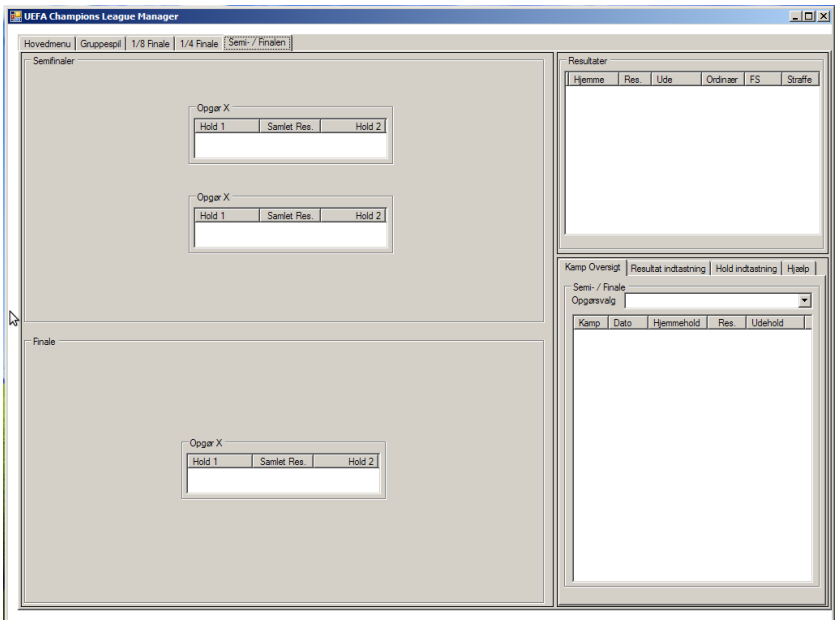
Figur A.28: Gruppespilsskærm.



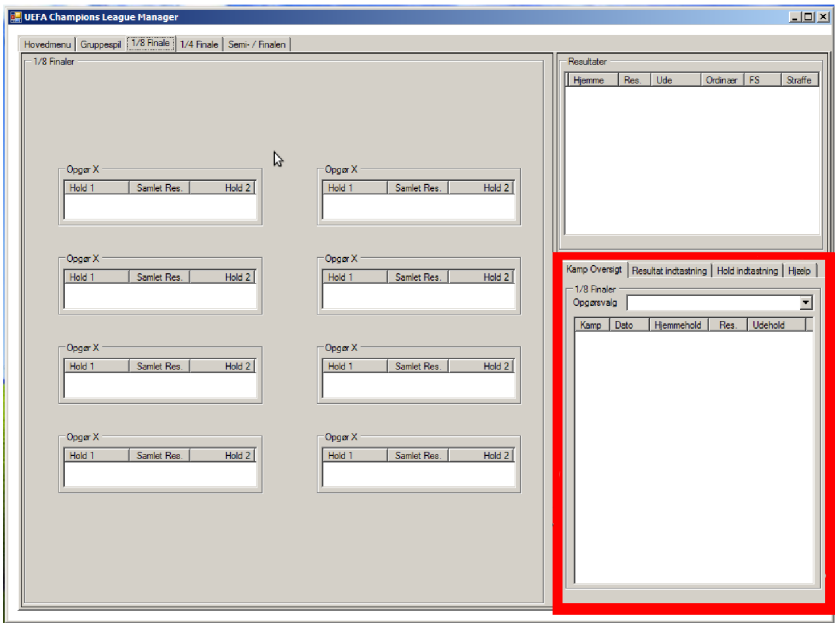
Figur A.29: Ottendedelsfinaleskærm.



Figur A.30: Kvartfinaleskærm.



Figur A.31: Semi- og finaleskærm.



Figur A.32: I den røde ramme ses fanebladene til systemets administrative funktioner.

A.2.9 Den tekniske platform

Systemet udvikles i C# og skal kunne afvikles på en PC, med Windows XP SP3 som styresystem. Ydermere skal der være installeret .NET 3.5 Framework. Man skal anvende en vindues-/fanebladsbaseret brugergrænseflade og skal styres med almindelig mus og tastatur.

A.2.10 Anbefalinger

Systemets nytte

Vores design imødekommer de væsenligste kriterier, med følgende bemærkninger:

- Brugbarhed
  Dette bliver vurderet ved en aftestning af den første version af systemet.

- Flytbarhed
  Systemet er lavet til at kunne flyttes, idet det ikke er baseret på databaser. Man kan eventuelt lave programmet så det kan ligge på en USB lagerenhed, så det kan flyttes efter ønske. Vi ser det ikke realiserbart at UEFA vil bruge vores system. Derfor ser vi ikke nogen større mulig udbredelse.

- Korrekthed og forståelighed
  Et review udført den 11. maj 2009 konkluderer, om designet opfylder disse kriterier.

Plan for ibrugtagning

Vi forventer ikke at UEFA vil bruge vores system, men skulle det mod forventning ske, skal det sendes på en CD-ROM eller en USB lagerenhed. Vi skal også afklare med UEFA, hvem der står for vedligeholdelse.

Implementeringsplan

Vi står selv for hele udviklingen af programmet, idet det er en del af vores projekt. Dette vil blive suppleret med support fra folk med kyndig viden omkring programudviklingsområdet. Vi vuderer at systemet kan realiseres, på baggrund af en gruppe på 4-6 personer, der er på 2. semester i BAIT uddannelsen. De skal arbejde på projektet i 2-3 måneder, herunder inklusiv programevaluering, før det er klar til brug.

A.3 Programtest

A.3.1 Formål

Denne test har til formål at identificere eventuelle problemer, der måtte være i opbygningen af programmet. Den vil være opbygget som en såkaldt "black box" test, så vi kommer altså ikke ud i alle kroge af programmet, men holder os istedet til to klasser. Det er derfor vigtigt, at lave et forarbejde, for at finde eventuelle svage punkter i programmet.
Disse svage punkter kunne eventuelt være, at det er muligt at indtaste forkerte tal (for eksempel negative i stedet for positive tal). Dette kan medføre, at man får et forkert output fra programmet, idet det ikke ved, at negative tal ikke må indtastes.
Det er derfor vigtigt at få udbedret fejlen, hvis vi identificerer et sådant svagt punkt.
Denne test fokuserer primært på metoder, da vi finder det mest sandsynligt, at fejl kunne opstå i disse. Vi benytter NUnit Framework til at teste og Doxygen til at dokumentere testen.

A.3.2 Identificering af elementer i klasserne.

I alle klasser findes der elementer. Disse går blandt andet under navnene variable, konstruktorer, properties og metoder.
I følgende afsnit opridser og identificerer vi de forskellige elementer i hver klasse. Derudover har vi valgt at opridse, hvad der kunne være interessant at kigge på, i forbindelse med selve testen.

Klassen Kamp:

I klassen Kamp er det vigtigt at teste om diverse properties virker, da det er interessant at finde ud af om de giver den forventede værdi tilbage.
Derudover er det vigtigt at finde ud af, om metoden TildelPoint fungerer korrekt.

Variable	Klassen Kamp indkapsler fire variabler: int? hjemmemaal, udemaal; Hold hjemme, ude;
Konstruktorer	Klassen Kamp indeholder en konstruktor, der initialiserer de fire variabler. (Hold hjemme, ude;) Hold får får tildelt et ude og et hjemmehold i kontruktoren (int? hjemmemaal, udemaal;) Bliver begge sat til at have en Null værdi
Properties	Klassen indeholder fire get/set properties, en for hver variabel.
Metoder	Klassen indeholder metoden TildelPoint, der tager to argumenter(int hjemmemol, int udemol). Metoden sammenligner de to resultater og tildeler hhv. 3, 1 eller 0 point til de enkelte hold i kampen. Dernæst lægger den +1 spillet kamp til holdet. Derudover giver den hjemmemaal og udemaal en ny værdi.
Andet	Klassen indeholder også en simpel textoutput, der udskriver hvilke hold, der spillede kampen og hvad resultatet blev.

Klassen KnockoutKamp:

I klassen KnockoutKamp er det relevant, at finde ud af om metoden FindVinder returnerer den korrekte vinder. Dette er gjort ud fra resultaterne af de to kampe, samt forlænget spilletid og straffesparkskonkurrence.

Variable	Klassen KnockoutKamp indkapsler otte variable: private int? vinderfsmaal; private int? andenpladsfsmaal; private int? vinderstraffe; private int? andenpladsstraffe; private Hold vinder; private Hold andenplads; private Kamp ko1; private Kamp ko2;
Konstruktor	Klassen har en konstruktor, der tager to hold og sætter dem til at være henholdsvis vinder og andenplads. Derudover initialiserer den to kampe(i en knockoutkamp) Ko1 og Ko2
Properties	Klassen indeholder get/set funktioner for alle variablerne.
Metoder	Klassen indeholder to væsentlige metoder, UdskrivStilling og FindVinder. Den første metode lægger de to kampes målscorer for henholdsvis Vinder og Andenplads hold sammen og udskriver den. Den anden metode indeholder en algoritme, der finder vinderen ud fra ordinær spilletidsmål, forlænget spilletid og straffesparksmål.

A.3.3 Opbygning af test

Kamp

I vores program kalder vi testen af Kamp for Kamptest.
I klassen Kamp vil vi fortrinsvis beskæftige os med metoden TildelPoint, idet det er den vigtigste del af klassen.
Vi vil lave en test, der kontrollerer et holds navn. Da dette navn er knyttet til et hold ved initialiseringen, er det interessant at finde ud af om navnene passer til holdet.
Dernæst vil vi teste om hjemme- og udeholdet passer til holdene. Idet der er knyttet to hold til en kamp, er det vigtigt at finde ud af om henholdsvis Hjemmehold og udehold, er knyttet til de rigtige hold.
Vi vil også kigge på metoden TildelPoint. Her er det især interessant, at se om metoden tildeler 3, 1 eller 0 point alt efter vundet, uafgjort eller tabt kamp.

Knockoutkamp

I vores program kalder vi testen af KnockoutKamp for Knockkamptest.
Her vil vil teste de to metoder UdskrivStilling og FindVinder.
Ligesom klassen Kamp, vil vi igen teste om hjemmehold og udehold, er blevet initialiseret korrekt og tildelt de korrekte hold.
Vi vil lave en test, der kontrollerer metoden UdskrivStilling. Metoden skulle gerne returnere en tekststreng, hvor målene for hvert hold er talt sammen.
Vi vil også teste metoden FindVinder. Denne metode er især vigtig, fordi den sammenligner og returnerer en vinder af en knockoutkamp. Der kontrolleres, at hvis andenpladsmål er lig vinderens mål (sammenlagt), så kigges der på udemålene. Hvis disse også er lig hinanden, springes der videre til FSmaal, hvilket vil sige forlænget spilletid. Her kigges der på, hvem der har flest mål i forlænget spilletid. Hvis disse også er lig hinanden, springes der videre til straffesparksmål. Igen kigges der på hvem der har flest mål her. Hvis disse også er lig hinanden (hvilket dog ikke burde kunne ske, idet kampen ikke er spillet), skulle null gerne returneres.

A.3.4 Resultat af test

I vores tests af klassen Kamp og KnockoutKamp, fandt vi 1 fejl. Vores test af klassen KnockoutKamp viste det sig, at den sorteringsalgoritme vi oprindeligt havde tiltænkt skulle finde en vinder, ville give en fejl, hvis der var lige mange straffespark indtastet. Det vil sige, at hvis der skulle ske en indtastningsfejl, så ville algoritmen returnere hold 2. Den oprindelige algoritme, som ses i kodestykket herunder, skulle naturligvis have returneret et tomt hold. Dette har vi nu rettet i vores kode.

```
public Hold findvinder()

{

 if ( ko1.Hjemmemaal + ko2.Udemaal > ko1.Udemaal + ko2.Hjemmemaal)
     return Andenplads;

 else if (ko1.Hjemmemaal + ko2.Udemaal < ko1.Udemaal + ko2.Hjemmemaal)
     return Vinder;

 else //if (ko1.Hjemmemaal + ko2.Udemaal == ko1.Udemaal + ko2.Hjemmemaal)

 if (ko1.Udemaal > ko2.Udemaal)
     return Vinder;

 else if (ko1.Udemaal < ko2.Udemaal)
     return Andenplads;

 else //if (ko1.Udemaal == ko2.Udemaal)

 //der skal smides noget forlænget spilletids resultat input ind her
 if (VinderFSmaal > AndenpladsFSmaal)
     return Vinder;

 else if (VinderFSmaal < AndenpladsFSmaal)
     return Andenplads;

 else //if (VinderFSmaal == AndenpladsFSmaal)


 if (VinderStraffe > AndenpladsStraffe)
     return Vinder;

 else (VinderStraffe < AndenpladsStraffe)
     return Andenplads;

       }
```

A.4 Usabilityevaluering

A.4.1 Formål

Som et led i vores programmering af brugergrænsefladen, hører en usabilitytest til. Denne test har til formål, at afsløre eventuelle fejl, der måtte være i systemets brugbarhed.

I det følgende beskriver vi, hvordan usabilitytesten skal udføres. Testen vil blive afholdt i vores grupperum på universitetet. Testen skal udføres, som en såkaldt "tænke-højt test". Det vil sige, at personen siger hvad han/hun gør, inde i programmet. For at kunne få maksimalt udbytte af dataene, vil der bagefter blive afholdt et uformelt interview. Denne har til formål at belyse, hvilke dele af programmet, vores testperson finder ubrugbare.
Vores testperson er Ditte Hertzum på 23 år, der til dagligt læser til bibliotekar. Hun er vant til at have med administrative programmer at gøre, derfor har vi netop valgt hende. Der vil i underafsnittene til afsnittet "usabilitytesten" herunder, være udførlige guides til testlederen (introduktion og formelt interview), samt til testpersonen.

A.4.2 Evalueringen

Indledning

Gruppespillet og 1/8-finalerne, er blevet valgt som emner for usability testen. Disse er blevet valgt, da det er to kernedele af programmet. Man kunne også have testet 1/4-finalerne, semifinalerne og finalen. Disse er ikke blevet valgt til test. Grunden til dette er, at 1/4- og semifinalerne ligner 1/8-finalerne. Den eneste forskel er antallet af opgør. Finalen er den eneste, der kodemæssigt, er bygget anderledes op end de andre, men denne var ikke færdigudviklet på det givne tidspunkt, for usability testen.
I både gruppespillet og 1/8 finalen, er opgaverne blevet konstrueret til, at teste både indtastning og aflæsning. I indtastningsopgaverne bliver der testet for både holdindtastning og resultatindtastning. Aflæsningsopgaverne er en del mere simple og testpersonen skal kun lokalisere en oversigt over turneringen (gruppespillet og 1/8 finalen) og derefter trykke på Print Screen.

Udstyr

For at dokumentere vores test skal vi bruge en PC med en standard Windows XP installation, der opfylder de fastsatte krav for afvikling af vores system. Disse er nævnt i afsnittet om den tekniske platform i analysedokumentet (se side 49). Derudover skal vi bruge software til at optage det skærmbillede testpersonen ser, til senere evaluering af testen. Til dette har vi valgt freeware programmet CamStudio [RenderSoft Software, 11. maj 2009].

Introduktion

Vi vil gerne takke dig for, at hjælpe os med at gennemføre denne test. Vi vil gerne gøre opmærksom på, at det er ikke dig vi vil teste, men selve programmet. Før testen skal jeg sige et par ting for at være sikker på, at det hele er på plads og at vi husker det hele.

Det system du skal hjælpe med at teste er et program som hedder UEFA Champions League Manager. Programmet skal hjælpe ansatte i UEFA (United European Football Association) med at holde styr på UEFA's Champions League, som er en fodbold turnering mellem europæiske klub hold. Testen vil foregå ved at du skal løse en række opgaver (eller så mange du kan nå). Opgaverne afspejler den tiltænkte brug af programmet. Du vil få disse opgaver udleveret en efter en.

Vi vil bede dig løse opgaverne på følgende måde:
Først læser du hele teksten til opgaven højt. Derefter fortæller du mig, hvordan du forstår opgaven, og hvad du vil gøre for at løse den. Så går du i gang med selve løsningen. Mens du arbejder med opgaven vil vi gerne have, at du tænker højt. Det betyder, at du skal sige, hvad du har tænkt dig at gøre, hvad der overrasker dig og hvad du ellers tænker på under brugen. Jeg ved godt, at det ikke er naturligt at sidde og tale højt om sine tanker og bevægelser, så hvis du glemmer det, vil jeg huske dig på det.
Hvis du får problemer under løsningen af opgaven, kan du godt spørge mig. Men jeg vil nok ikke hjælpe dig direkte. Jeg vil i stedet forsøge at få dig til selv at komme videre.

Når du mener, at du er færdig med at løse en opgave, vil jeg bede dig sige det, så vi ikke er i tvivl bagefter. Før testen starter, vil jeg bede dig om at underskrive denne samtykkeerklæring for at sikre, at du er indforstået med rammerne for testen.

Opgaver

UEFA's Champions League består af indledende runder, hvor de forskellige klub hold spiller om pladser til gruppespillet. I gruppespillet er der 8 grupper med hver 4 hold. I hver gruppe bliver der spillet 12 kampe, 6 kampe per hold. Når gruppespillet er færdigt vil første og anden pladsen i gruppespillet gå videre til knockoutrunderne/finalespillet i første omgang 1/8 finalen. Her gælder spiller en første og en andenplads mod hinanden i et opgør, som består af to kampe en på hjemmebane og en på udebane. Vinderen af dette opgør går videre til 1/4 finalen. I 1/4 og semifinalen foregår på samme måde som 1/8 finalen. I finalen spiller vinderne af semifinalen mod hinanden og vinderen af denne kamp vinder Champions League. Her bliver der kun spillet en kamp.

Opgave 1A

Du er lige blevet ansat af UEFA til administrere Champions League. Du har fået et papir i hånden, hvor der en liste med grupper A-H og under hver gruppe er der listet nogle hold. Din chef har bedt dig om at indtaste holdene til gruppespillet i deres nye program UEFA Champions League
Manager. Du ser på klokken og indser du lige kan nå at indtaste 2 grupper inden frokost. Holdene er som følger.

Gruppe A:

- Manchester United
- Aalborg Boldklub
- FC Porto
- Werder Brehmen

Gruppe B:

- Ajax Amsterdam
- Manchester City
- Real Madrid
- Newcastle

1B

Da du kommer tilbage fra frokost opdager du, at du har indtastet et hold forkert. Gå tilbage til gruppe A og ændre FC Porto til Villareal.

1C

Du kommer pludseligt i tanke om, at din chef også sagde, at holdene skulle indtastes præcis som de stod på papiret, ellers ville gruppespillet ikke blive oprettet rigtigt. Du beslutter dig for at dobbelt tjekke dine indtastninger og opdager pludseligt, at Manchester City skulle stå øverst og Ajax Amsterdam nederst i gruppe B. Du skal ændre disse så gruppen bliver rigtig.

1D

Du sidder og kigger på klokken, der er kun en time til fyraften og du glæder dig til at komme hjem. Pludseligt kommer din chef ind med en ny liste fuld af resultater fra kampe i gruppespillet. Han beder dig om, at skrive resultaterne ind i programmet. Listen er som følger.

Gruppe A:

- 6/5-2009 - Aalborg Boldklub 1-1 Villareal
- 6/5-2009 - Werder Brehmen 1-2 Manchester United
- 11/5-2009 - Manchester United 1-3 Aalborg Boldklub
- 11/5-2009 - Villareal 1-1 Werder Brehmen

1E

10 minutter inden fyraften er du færdig med indtastningerne og du tænker, at du nok hellere må kontrollere indtastningerne igen. Du fanger en lille fejl i resultatet mellem

- 11/5-2009 - Manchester United -Aalborg Boldklub

Resultatet skulle have været 2-3, ret og gem resultatet.

1F

Næste dag vil chefen gerne have at vide, hvordan point stillingen er i gruppespillet. Find point- stillingen over hele gruppespillet og tryk Print Screen(på tastaturet).

2A

Efter nogle måneders arbejde hos UEFA er Champions League nået til 1/8 finalen. Din chef vil godt have dig til at indtaste de 8 opgør som udgør 1/8 finalen. Opgørende er som følgende.

- Ajax Amsterdam - Manchester City
- FC Barcelona - Manchester United
- Sevillia - Westham United
- Valencia - Brøndby IF

- Rhyl FC - FC København
- Newton AFC - FC Köln
- FC Midtjylland - Liverpool
- Aalborg Boldklub - Chelsey

2B

Chefen synes du har gjort et godt stykke arbejde og vil give dig tidligt fri, men du skal lige indtaste resultaterne i 1/8 finalen først. De er som følger.

- Ajax Amsterdam 1-1 Manchester City
- Manchester City 2-1 Ajax Amterdam
- FC Barcelona 3-2 Manchester United
- Manchester United 4-1 FC Barcelona
- Sevillia 0-0 Westham United
- Westham United 4-0 Sevillia
- Valencia 0-8 Brøndby IF
- Brøndby IF 2-1 Valencia
- Rhyl FC 0-5 FC København
- FC København 3-2 Rhyl FC
- Newton AFC 1-1 FC Köln
- FC Köln 1-1 Newton AFC
- FC Midtjylland 5-2 Liverpool
- Liverpool 3-3 FC Midtjylland
- Aalborg Boldklub 8-0 Chelsey
- Chelsey 0-8 Aalborg Boldklub

2C

Du vil gerne vise din chef det flotte stykke arbejde du har lavet (så du endeligt kan komme hjem). Find oversigten over de 8 opgør i 1/8 Finalen og tryk Print Screen (På tastaturet).

A.4.3 Resultater af usabilityevaluering

Indledning

I det følgende bliver de forskellige problemer, der blev fundet ved usabilitytesten, opremset. Først problemerne fra gruppespillet, dernæst dem fra 1/8 finalerne.

Problemer under gruppespil

Nr.	Problem	Klassificering
1	Havde problemer med at identificere Gruppe A i gruppeoversigten og Gruppe A i hold indtast fanebladet, fordi der ikke var konsistens mellem indtastede data. *Mangler konsistens mellem data i indtastningsfanebladet i gruppeoversigten. Holdene kommer ikke i samme rækkefølge begge steder. Dette er dog en vigtig ting da holdene skal indtastes i bestemt rækkefølge for at kamplanen bliver generet korrekt.*	**Alvorlig:** Kategoriseres alvorlig, da det kan skabe forvirring om de indtastede hold, er korrekt indtastet. Dette kan gøre det nødvendigt for brugeren at kontrollere indtastninger igen, eller skabe unødvendigt forvirring om, hvilke data er korrekte.
2	Tror hun skal indtaste direkte i gruppeboksen i gruppeoversigten. *Indtastningsområde kunne gøres tydeligere.*	**Kosmetisk:** Kategoriseres kosmetisk, da de ikke har større indvirkning på brugen af programmet og efter forholdsvis kort tid finder brugeren ud af, hvor man indtaster data og denne fejl forsvinder.
3	Prøver om "Enter"-knappen virker ved indtastning af hold *Mangler "Enter" funktion ved indtastning af hold, så man ikke behøves at bruge musen til, at tilføje hold med.*	**Kosmetisk:** Kategoriseres som kosmetisk, da dette ikke har nogen betydning for indtastning af hold og opholder heller ikke testpersonen i længere tid.
4	Har problemer med at finde ud af, hvordan man ændrer et hold navn. *Det er ikke klart, at man kan ændre holdnavne direkte i holdindtastningsfanebladet, ved at slette et hold og indskrive et nyt.*	**Alvorlig:** Kategoriseres alvorlig, da det tager testpersonen længere tid om at finde ud af, hvordan man skal gøre for at indtaste et andet hold.

5	Slet knappen i holdindtastning i gruppespillet virker ikke *"Slet"-knappen under holdindtastning i gruppespillet virker ikke, dette indvirker i at den stillede opgave ikke kan gennemføres og testpersonen bliver nødt til at gå videre.*	**Kritisk:** Kategoriseres kritisk, da det betyder at gruppen i gruppespillet ikke kan indtastes korrekt, hvilket igen betyder at resten af turneringen muligvis ikke kan oprettes korrekt. I værste fald skal programmet genstartes for at kunne rette fejlen, dette ville spilde unødigt meget tid.
6	Testpersonen har svært ved at se om gruppeoversigten bliver opdateret, efter at have flyttet rundt på hold i holdindtastningen og hun har trykket gem. *Gruppespillet bliver ikke opdateret i konsistens med gruppeindtastningsskærmen og Resultatoversigtsskærmen.*	**Alvorligt:** Kategoriseres alvorligt, da det skaber unødigt meget forvirring for testpersonen. De tre dele (gruppeoversigten, resultatoversigten og holdindtastning), bliver ikke opdateret konsistent og alle hold står forskelligt alle tre steder. Dette kan i værste fald få brugeren til at bruge unødigt lang tid, på at kontrollere indtastningerne. Testpersonen kan dog ikke være sikker på, hvilke data der er korrekt indtastet, da alle skærmene viser noget forskelligt. (Problemet opstår når der ikke er indtastet resultater).
7	Testpersonen prøver at indtaste point, direkte i gruppeoversigtens gruppebokse. *Det virker naturligt for testpersonen, at indtaste data direkte i gruppeboksene i gruppeoversigten, dette kan dog ikke lade sig gøre.*	**Kosmetisk:** Kategoriseres kosmetisk, da det ikke opholder testpersonen i længere tid. Efter kort tid finder hun det rigtige sted, at indtaste data og dette ser ikke længere ud til at være et problem. Kunne tænkes ikke at være et problem, hvis der var blevet givet undervisning i brug af programmet.

8	Efter at have indtastet et resultat i en kamp i gruppespillet, kommer testpersonen til at dobbeltklikke på "gem resultat"-knappen. *Testpersonen opdager ikke, at dette medfører tilførselen af for mange mål og derfor ekstra point, til de hold der er blevet indtastet for.*	**Kritisk:** Kategoriseres kritisk, da man ved et uheld kan komme til, at tilføje for mange mål og point til en gruppe og nogle af dennes hold. Dette er et stort problem, da det i værste tilfælde kan medføre forkerte vindere af gruppespillet. Se også problem 9 i forlængelse af dette problem.
9	Testpersonen skal i opgaven ændre et resultat i en af gruppespillets kampe, dette bliver også gjort som tiltænkt, men testpersonen opdager ikke at der bliver tilført flere point til stillingen i gruppen, som ellers skulle have forblevet den samme. *Testpersonen opdager ikke, at det hun tror ændrer stillingen for den kamp, i virkeligheden tilføjer ekstra mål og point til de to pågældende hold.*	**Kritisk:** Kategoriseres kritisk, da man efter at have indtastet resultater i en gruppespilskamp, ikke længere kan ændre i de indtastede data. Dette vil betyde, at hvis man eventuelt kommer til at indtaste forkert data, da bliver man nødt til at lave hele turneringen om. Dette vil betyde meget unødvendigt tidsspild og det vil betyde, at man SKAL indtaste alle data i gruppespillets kampe korrekt, ellers vil gruppespillet ikke være korrekt. Der er ikke mulighed for at ændre eventuelle fejlindtastninger.

Problemer under 1/8 finalen

Nr.	Problem	Kategorisering
10	Testpersonen vælger 2 hold i holdindtastningen, men der bliver ikke valgt et "opgør", derefter trykker testperson på gem. *Efter at have trykket på gem fremkommer en fejlmeddelelse, hvilket får testpersonen til at gå i stå. Hun skal lige til at trykke "luk" (ved hjælp af rødt kryds i toppen af fejlmeddelelsen), men testleder siger at hun skal trykke på "Continue"-knappen i stedet.*	**Alvorlig/kritisk:** Kategoriseres alvorlig, da det kan koste ekstra tid, hvis testpersonen ikke ved, hvad der skal trykkes på. Kan dog også være kritisk, da programmet kan risikere at lukke ned ved tryk på luk program og risikere derved, at miste alt ikke gemt data i programmet.

11	Under indtastning af hold glemmer testpersonen at ændre opgør og får derved ændret de oprindelige hold til hold, der skulle have været i et andet opgør. *Testpersonen opdager først efter tredje indtastning, at hun har indtastet holdene i samme opgør og bliver derfor nødt til at indtaste holdene på ny.*	**Alvorlig:** Kategoriseres alvorlig, da dette tager unødvendigt lang tid. Kunne dog afhjælpes ved, at lave en popup boks der spurgte om man virkeligt ville ændre de indtastede hold i opgøret, hvis man var i gang med at overskrive disse.
12	Da testpersonen bliver bedt om at ændre i resultaterne, finder hun opgøret igen og er nødt til at indskrive begge kampe igen, for at ændre dette. *Resultaterne bliver ikke vist tekstboksene i resultatindtastningsskærmen, dette resultere i mindre spild af tid*	**Kosmetisk:** Kategoriseres kosmetisk, da dette ikke ser ud til at påvirke testperson overhovedet, men ville dog være en udmærket feature at have, hvilket i sidste ende ville spare tid, hvis man indtastede forkerte resultater.
13	Reset problem (måske gruppespillet) *Testpersonen prøver at slette holdets navn ved at trykke på res oppe i oversigten, fordi hun tror det betyder reset*	**Kosmetisk:** Idet testpersonen ikke har nogen reel viden om hvad res betyder, ved hun derfor ikke hvad knappen gør, fordi tooltip funktionen ikke virker, tager det hende længere tid og finde sletknappen.
14	Efter testpersonen har indtastet to kampe til 1 - 1, burde der have været indtastet forlænget spilletids resultat og måske straffesparks resultat. *Testpersonen ved ikke, at der skal indtastes resultater for forlænget spilletid og straffesparks konkurrence, så dette bliver ikke gjort. Det skal dog tilføjes, at dette heller ikke står i opgaven.*	**Alvorlig:** Kategoriseres alvorlig, da systemmet selv skulle vide, hvornår der skal indtastes resultater for forlænget spilletid og straffespark. Dette bliver dog ikke fundet af systemet og man kunne forestille sig, at der ikke blev indtastet rigtigt og systemet automatisk valgte en tilfældig vinder.

15	Da holdende ikke bliver sendt rigtigt videre til 1/8 finalen, får hun selv lov til at vælge hvem der skal spille denne. Hun vælger samme hold til at spille i flere opgør. *Der kan vælges et hold til at indgå i flere opgør, dette kan ikke lade sig gøre ifølge UEFA's regelsæt.*	**Kritisk:** Kategoriseres kritisk, da der ikke er oprettet fejlfinding på flere indtastninger af samme hold i 1/8 finalen. Hvis denne fejl ikke blev opdaget, kunne man få unødvendigt mange problemer med hele Champions League.
16	Da hun prøver at holde musen over Resultatskærmen, fremkommer der et "TESTING"-tooltip. *Forvirrer et kort øjeblik testperson, som forventer hjælp til resultatskærmen.*	**Kosmetisk:** Kategoriseres kosmetisk, da dette kun forvirrer testpersonen et kort øjeblik, dog ville det være en god ide at rette dette til brugbar information om resultatskærmen.
17	Bliver ikke nævnt af testpersonen, men der står u/a i mange af de felter, hvor der ikke er indtastet data. *Har umiddelbart ikke påvirket testpersonen i højere grad. Der skulle have stået "n/a" for "not available" i stedet for "u/a."*	**Kosmetisk:** Kategoriseres kosmetisk, da dette ikke har haft nogen indflydelse på testpersonen, man kunne dog forestille sige det ville skabe forvirring blandt rigtige UEFA ansatte da de ville bruge for eksempel resultatskærmen mere.
18	Bliver ikke nævnt af testpersonen, men man kan ikke se det fulde navn af de hold som man skal indtaste resultater for i 1/8 finalen. *Har umiddelbart ikke påvirket testpersonen i højere grad. Der skal være mulighed for at se hele navnet, ellers kan man forestille sig, at der kunne opstå forvirring ved indtastning. Dog kan holdende ses i opgørsvælgeren.*	**Kosmetisk:** Kategoriseres kosmetisk, da dette ikke har haft nogen indflydelse på testpersonen, man kunne dog forestille sig at det ville kunne skabe forvirring, hvis man kom til at fokusere på det.

Diskussion af problemer

Der er, som det ses i appendix A.4.3 side 76, (følgende problemer vil kunne findes i dette appendix), en del problemer i brugergrænsefladen. I det følgende afsnit vil de kritiske problemer blive afdækket og disse vil blive diskuteret.

Problem 5 er det første kritiske problem, der blev fundet i usabilityevalueringen og omhandler "slet"-knappen i holdindtastningen i gruppespillet. Det var lidt af en overraskelse at denne ikke virkede, især fordi den havde virket tidligere i program udviklingen. Men på grund af ændringer i programmet og manglende evaluering af disse ændringer, virkede "slet"-knappen ikke under evalueringen. Dette var meget uheldigt, især da testpersonen bliver nødt til at stoppe udførelsen af opgaven.

Det var dog endnu mere uheldigt, at de data der skulle have været indtastet under opgaven, skulle bruges i senere opgaver. Selvom at dette var meget uheldigt under evalueringen, giver det dog et godt indblik i, hvordan mindre ændringer i koden, kan forårsage større problemer for hele programmet. Men viser også, at det er vigtigt at få evalueret programmet af andre end udviklerne, da disse kan tro at programmet virker korrekt og undlader at evaluere funktioner, som umiddelbart ser ud til at virke.

Man kan umiddelbart tænke, at en "slet"-knap der ikke virker, ikke kan have den store indflydelse på programmet. Men i dette tilfælde, viste det sig dog at have større konsekvenser, idet en forkert indtastning af hold ville betyde, at man skulle genstarte programmet og indtaste hold helt forfra.

Problem 8 og 9 omhandler samme problem, nemlig at man i resultatindtastningsskærmen kan indtaste et resultat og ved gentagende tryk på gem knappen, tilføjer flere point og mål til de hold, der bliver indtastet resultater for. Dette er endnu et stort problem, da testpersonen ikke opdager at der bliver tilføjet flere point til gruppen, kunne man forestille sig, at en UEFA medarbejder heller ikkSoftwaree ville opdage det.
Dette betyder, at der kommer fejl i gruppen og dette kan i værste tilfælde betyde, at de forkerte hold bliver overført til 1/8 finalen. Men disse problemer er også interessante at tage fat i. De viser nemlig at det er en god ting at lave en usabilityevaluering, da det ikSoftwareke kun er deciderede brugbarhedsproblemer man finder, men man kan også finde problemer i de metoder programmet indeholder.

Problem 10 omhandler en fejl, der fremkommer når man i 1/8 finalen prøver at tildele to hold til et opgør, men glemmer at vælge et opgør. Fejlen er en 'unhandled exception', fordi den ikke kan håndtere indekset fra listboxen. I evalueringen gav den mulighed for at fortsætte eller lukke programmet. Man kan dog ikke bero sig på, at kunne klikke sig videre og programmet går altså ikke ned, men fejlen er stadig kritisk og den skal derfor rettes. Her kan man igen se at evalueringen har været effektiv, da man som udvikler ikke nødvendigvis ville lave denne fejl, da man ved præcis hvad man skal gøre, når man indtaster data.

Problem 15 viser sig når man skal indtaste hold i 1/8 finalen. Her kan man vælge det samme hold, til at spille flere opgør. Dette gør at man kan risikere at opstille en forkert 1/8 finale. Denne fejl blev egentligt opdaget ved et tilfælde, idet holdene fra gruppespillet ikke var blevet overført rigtigt, fik testpersonen "frie tøjler" til at vælge nogle hold selv. Dette betød imidlertid, at hun valgte det samme hold for flere af opgørene.
Det er interessant at tænke på, da man måske skulle overveje ikke at lave helt fastsatte opgaver, eller man burde måske overveje opgaver, hvor testpersonen selv fik lov til at vælge, hvad han eller hun ville gøre. Dette kan ikke lade sig gøre i alle opgaver, men man kunne da overveje nogle situationer, hvor dette kunne være en interessant løsning, idet man i denne opgave fandt en fejl, man måske ellers ikke ville have opdaget.

Problemløsning

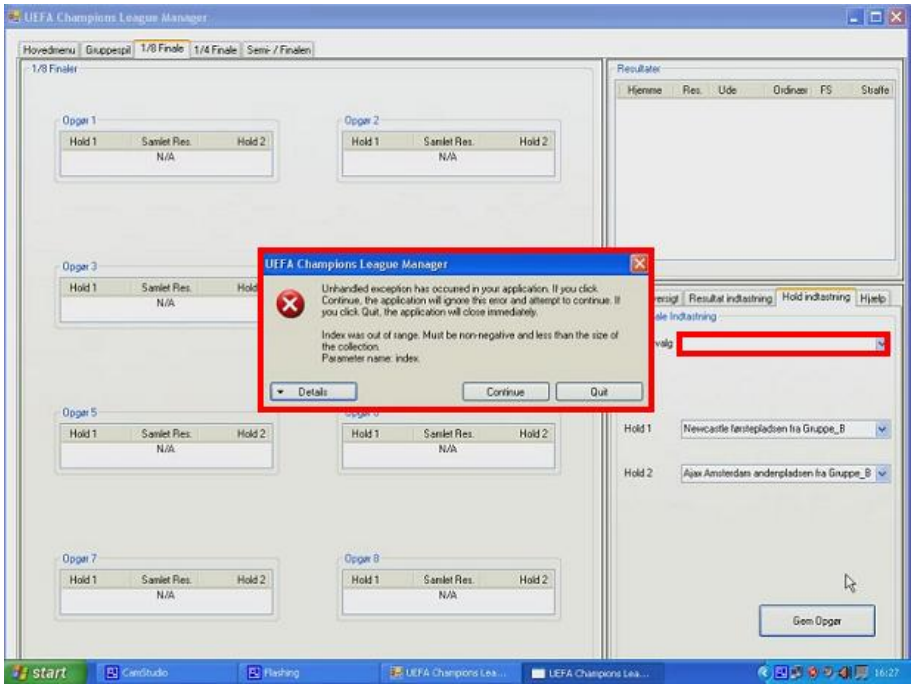
Efter usability testen er det blevet diskuteret, hvilke af problemerne der skal rettes. På nuværende tidspunkt er der ikke tid til at ordne alle problemerne, så de mest presserende er blevet valgt til at skulle rettes. Derudover vil nogle af de kosmetiske også blive rettet, da disse ikke er særligt tidskrævende. I efterfølgende afsnit vil der blive opstillet nogle af de problemer, vi har valgt at rette. Der vil også blive diskuteret, hvad der skal til for at rette disse. Alle problemerne kan findes i problemlisten, se A.4.3 side 76. Der vil foreligge en kort forklaring på, hvorfor det er et problem, samt hvad løsningen på dette er.

- Problem 3 - "Enter"-knappen
  Er blevet valgt da det drejer sig om en "Enter" funktionalitet på en tekstboks, som skal overføre noget tekst til en listboks. Det er ikke et problem der ødelægger funktionaliteten af programmet, men det vil øge brugbarheden, ved at man undgår at tage fat i musen. Da det er en funktion man skal bruge flere gange, vil det være en god tilføjelse til programmet. Løsningen af problemet blev at tilføje et event der reagerer på tryk af "Enter"-knappen. Dette event kalder herefter samme funktion som "Tilføj Hold"-knappen.

- Problem 5 - "Slet"-knappen
  Er blevet valgt da det er et kritisk problem, som kan kan forårsage at man, som det er nu, bliver nødt til at genstarte programmet. Ydermere kræver det at man skal indtaste holdene korrekt, da det ikke vil være muligt at bruge slet knappen. Man kan ikke tillade sådan et problem i den færdige version af programmet og det har derfor høj prioritet til at blive rettet. Under programmeringen er eventets navn blevet ændret. Der skulle således blot tilføjes nye referencer i programmets designklasse.

- Problem 10 - Fejlmeddelelse
  På figur A.33, side 84, kan man se den fejlmeddelelse, der fremkommer, når man prøver at gemme to hold i et tomt opgør. Fejlmeddelelsen, der fremkommer, er en Unhandled Exception, der opstår, fordi man ikke kan vælge indeks -1 i en liste, der starter fra nul. Når dette prøves kommer der en popup, som spørger om man vil fortsætte eller lukke programmet. Hvis der vælges fortsæt, kan dette lade sig gøre. Det er dog ikke hensigtsmæssigt at have sådanne fejl i et program. Det har derfor fået høj prioritet til at blive rettet. Løsningen af problemet blev at ændre på "Enabled propertien" således at hold indtastningens felterne ikke er brugbare, før et opgør er valgt.

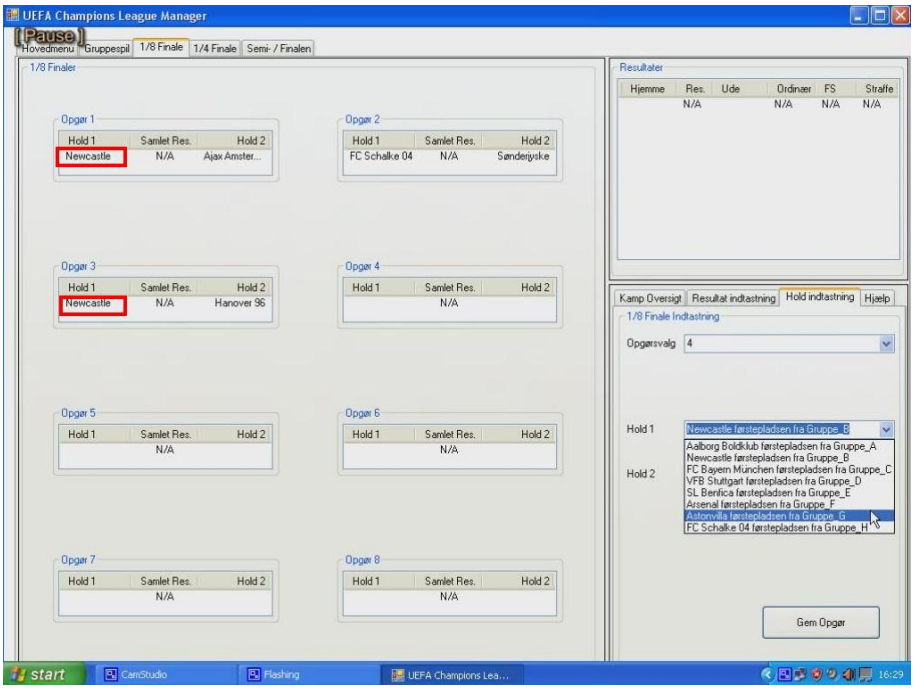
- Problem 11 - Holdindtastning (1/8 finale)
  Er manglende information, hvis man prøver at indtaste hold to gange i samme opgør i 1/8 finalen. Dette er også et brugbarhedsproblem og vil kunne løses ved at tilføje en popup besked, når man prøver at indtaste andre hold. Løsningen af problemet blev at gennemløbe alle kampe i variablen Ottendedelsfinale, for herefter at fjerne de hold, som allerede er tilskrevet en kamp. Herved kan holdene kun vælges en gang.



Figur A.33: Skærmbillede af problem 10. Fejlmeddelelsen og den tomme listbox, der forårsager den, er fremhævet med rødt.

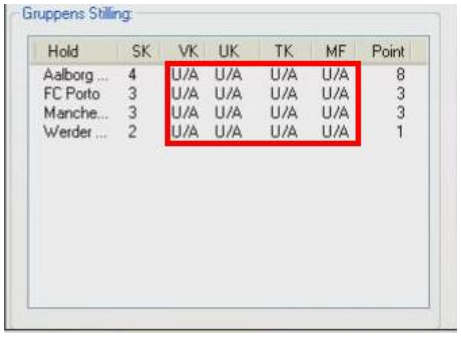
- Problem 15 - Holdindtastning (1/8 Finale)
  Er endnu et kritisk problem, der kan ses på figur A.34, side 85. I 1/8 finalen kan man indtaste det samme hold i flere opgør, så runden bliver oprettet forkert. Dette kan afhjælpes ved at fjerne holdene fra de to lister, som bruges til at vælge hold til opgørene. Det vil sige hver gang man har gemt to hold forsvinder de fra listen, så man ikke kan vælge dem igen. En anden metode er hvis programmet gør opmærksom på, at man har valgt det samme hold flere gange. Problemet er blevet fjernet med løsningen på problem 11.

- Problem 16 - Tooltip
  Er et "tooltip" som fremkommer, hvis man holder musen over resultatskærmen (øverst til højre i 1/8 finalen). Her kan man vælge at fjerne "tooltippet" helt eller for at øge brugbarheden af resultatskærmen, kan man lave en oversættelse af de forkortelser der som knytter sig til den. Dette har også været meningen fra starten, men er blevet nedprioriteret i forhold til vigtigere programmeringsmæssige aspekter.
  Teksten i tooltippet er blevet ændret til at dække hele teksten. Det har ikke været muligt for os at finde en løsning på at lave tooltips på hver enkelt kolonne i vores Listviews, derfor har vi lavet et samlet tooltip, der beskriver de forskellige forkortelser.



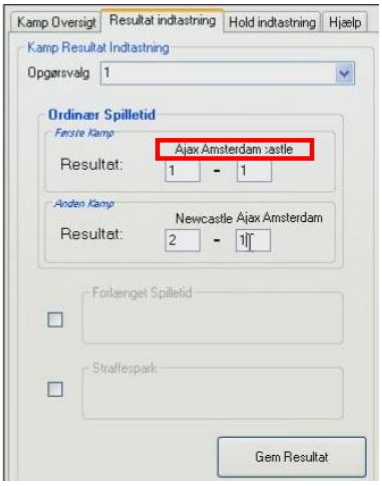
Figur A.34: Skærmbillede af problem 15. Det samme hold, der er markeret med rødt, er blevet indtastet flere gange.

- Problem 17 - Listview (N/A)
  Problemet er vist på figur A.35. Det er et meget nemt problem at løse, da det blot drejer sig om tekst i nogle af de listview, der er blevet brugt, hvor der står u/a, men der skulle have stået n/a.



Figur A.35: Skærmbillede af problem 17. Detalje informationerne, der har en forkert betegnelse er fremhævet med rødt.

- Problem 18 - Tekst i labels
  Er endnu et af de nemme problemer at løse, hvor teksten overlapper hinanden i resultatindtastningsskærmen, som vist på figur A.36. Dette kan løses ved at sætte tekstens udbredelses retning anderledes, for eksempel fra højre mod venstre. Tekst alignment properties er blevet ændret, således at teksten nu knytter sig til højre side af boksen. Ydermere er Autosize blevet slået fra og labels størrelse manuelt bestemt.



Figur A.36: Skærmbillede af problem 18. Holdnavnene der er fremhævet med rødt overlapper hinanden.

Efter gennemgang af ovenstående fejl, har vi konstateret følgende nye fejl, som de mest markante, der bør rettes i næste revidering. Disse skal tilføjes til de fejl, der ikke er blevet rettet.

- Mangler en advarsel mod dobbelt indtastning af resultater i både gruppe- og finalespillet.
- Tooltip skal rettes til i hele programmet.
- Opstartstid på introfilm.
  - Skal længere forsinkelse på denne.
  - Teksten skal rettes til.
  - Opstartstid af introfilm skal kortes ned, måske med mulighed for at slå den fra.
- Opdatering af vinduer/tabs skal rettes til, da der er problemer med at vise opdaterede data.
- Listen i kvartfinalen opdateres ikke ved valg af kamp og resulterer i nogle finaler.
- Opgørsvælger i kvartfinalen/semifinalen, skal flyttes nærmere hinanden sådan at det virker mere intuitivt at vælge opgør inden man vælger hold. Hvis det skal kobles sammen med ovenstående, kunne det være en idé at advare mod et allerede oprettet opgør på pladsen.

A.5 Dokumentation af programmet

A.5.1 Henvisning til programdokumentation

Dokumentation af programmet kan ses på den medfølgende CD. Alt den vedlagte dokumentation, er blevet til ved hjælp af Doxygen, som er et hjælpe program til at organisere det dokumentation, der er skrevet ind i programkoden. Denne bliver så vist på en let og overskuelig måde.