STINE ANDERSEN, ANDREAS LØFGREN & JIMMI SCHÆFFER

UNDERVISERE: David Stougaard, Marc Halgreen & Anette Tjørnemark

EKsamensopgave 1 semester - datamatiker

Fodboldturnering

Indhold

[IT i Organisation 4](#_Toc405721299)

[Organisatorisk beskrivelse 4](#_Toc405721300)

[Overvejelse om decentralisering 4](#_Toc405721301)

[Fordele og ulemper ved et digitalt system 5](#_Toc405721302)

[Fordele ved et digitalt system 5](#_Toc405721303)

[Ulemper ved et digitalt system: 6](#_Toc405721304)

[Personlighedstræk hos bruger 6](#_Toc405721305)

[Udfordring af efterspurgte IT-løsning 8](#_Toc405721306)

[Systemudvikling 12](#_Toc405721307)

[Kravspecifikation fra systembruger/køber 12](#_Toc405721308)

[Mindstekrav: 12](#_Toc405721309)

[Konkurrenceparametre: 13](#_Toc405721310)

[Domænemodel 13](#_Toc405721311)

[Anvendelsesområder 15](#_Toc405721312)

[Brugsmønster 1 16](#_Toc405721313)

[Brugsmønster 2 17](#_Toc405721314)

[Brugsmønster 3 18](#_Toc405721315)

[Brugsmønster 4 18](#_Toc405721316)

[Brugsmønster 5 19](#_Toc405721317)

[Designmodel 19](#_Toc405721318)

[Planlægning af projektet 20](#_Toc405721319)

[Programmering 24](#_Toc405721320)

[Beskrivelse af programmet 25](#_Toc405721321)

[Forbedringer 27](#_Toc405721322)

[Exceptionhandling 27](#_Toc405721323)

[Test 27](#_Toc405721324)

[Database 28](#_Toc405721325)

[Konverterings algoritmen 28](#_Toc405721326)

[Normalisering 29](#_Toc405721327)

[Første normalform 29](#_Toc405721328)

[Anden normalform 29](#_Toc405721329)

[Tredje normalform 30](#_Toc405721330)

[Bilag 1 31](#_Toc405721331)

[Database 31](#_Toc405721332)

[Bilag 2 36](#_Toc405721333)

[Opgaveformulering 36](#_Toc405721334)

[Beskrivelse af program 42](#_Toc405721335)

[Exceptionhandling 42](#_Toc405721336)

[Test 43](#_Toc405721337)

# IT i Organisation

## Organisatorisk beskrivelse

Sjællands Boldspil-Union (SBU) er opbygget af fodboldklubber på Sjælland og udgør derved en fælles union for fodbold på Sjælland. Unionen er organisatorisk underlagt Dansk Boldspil-Union (DBU) og derfor også Danmarks Idrætsforbund (DIF). Den er drevet af frivillige. Unionen har eksisteret i over 100 år. Kun i de seneste 15 år er ligatabeller blevet udgivet elektronisk, før det, blev ligatabellerne ført ved pen og papir.

Derfor at SBU nu bedt os om at komme med et bud på et system de kan benytte sig af, for at kunne holde bedre styr på alle de oplysninger og informationer der skal registerets igennem sæsonerne samt at kunne gemme disse oplysninger til fremtidige statistikker.

Turneringsledelsen er den administrative del af unionen der skal benytte sig af IT-løsningen. De sidder som sagt på den administrative del af unionen og har ansvaret for at bogføre kamprapporter, samt information af klubber og spillere og udgivelse af ligatal.

## Overvejelse om decentralisering

I unionen er der en stilling der sidder med ansvaret – det er turneringsledelsen.

Turneringsledelsen har det administrative ansvar – hvor de uddeler kampdage, samt holder styr på statistikker i form af målscoringer og ligatal, der tidligere blev udgivet elektronisk på internettet – samtidig har de også ansvaret for, hvis der skulle opstå problemer med fx en klub, der ikke kan spille en bestemt dato, så skal de kunne rykke kampen.

Derudover har medlemmer og spillere i organisationen mulighed for at bedømme dommerne og spillerne på deres præsentation fra eller under en kamp.

Derudfra kan vi konkludere at vi har en såkaldt flad organisationsstruktur, da der ikke er et stort hierarki af positioner og der er kort mellem dem der tager beslutninger og dem der udfører dem. Da vi kun har en gruppe der sidder med det administrative ansvar og ansvaret for organisationen – har vi derved en centralisering af ansvaret i organisationen og derved også en høj formalisering.

Når der er tale om en decentralisering, handler det om at vi uddeler ansvaret på en anden måde end den nuværende centralisering som ligger hos turneringsledelsen.

Derved vil vi overveje om vi kan implementere dommerne i IT-løsningen, så de har et ansvar i organisationen. Dette kan fx gøres ved, at vi giver dommerne mulighed for at kunne administrere eventuelle opståede klager omkring uddelte røde og gule kort samt at kunne flytte kampdatoer og tidspunkter.

På den måde kan vi uddele ansvaret i organisationen så alt information samt eventuelle ændringer ikke skal gå igennem turneringsledelsen. Samtidig kan vi bespare på mandetimer samt ressourcer i organisationen.

Dette vil også gøre formaliseringen lavere og give organisationen en fleksibilitet. Der opstår en lavere formalisering ved at der kommer et delt ansvar i organisationen, dette betyder at der er flere end en stilling der sidder med ansvaret i organisationen.

Denne beslutning vil dog ikke ændre i strukturformelen, da vi stadig ikke har særlig mange stillinger der deler ansvaret – der er stadig kun tale om 2 stillinger der skal deles om ansvaret. Dog ligger turneringsledelsen stadig som den øverste i strukturen fordi de er den øverste administrative stilling i organisationen.

## Fordele og ulemper ved et digitalt system

Når der skal ske en omvæltning af denne kaliber i organisationen, er vi nød til at kigge på de fordele og ulemper, der kan opstå ved denne digitalisering af organisationen, og derudfra at kunne bedømme hvorvidt det er muligt og om behovet mødes ved denne løsning.

### Fordele ved et digitalt system

* + **Tidsbesparende**: I vores problemstilling har vi fået oplyst at unionen er 100 år gammel, og at de kun de seneste 15 år har benyttet sig af at udgive ligatallene elektronisk. Derudfra kan vi konkludere at i de resterende 85 år, har unionen benyttet sig af pen og papir. Så vores tidsbesparing kan opnås ved at der bliver fremstillet et program hvori man både kan uploade og finde data om den valgte klub, spiller, dommer og kamp osv. samt finde en oversigt over de nævne data. Det tidsbesvarende ligger i at programmet allerede har alle informationerne omkring kampe, hold og spillere og selv opretter kampene, så turneringsledelsen skal mest sørge for at informationen er opdateret.
  + **Overblik**: Ved at vi får en IT-løsning implementeret til unionen, kan vi skabe et bedre og nemmere overblik, ved at give muligheden for at se en oversigt over diverse kampe, spillere og hold osv.
  + **Adgang til informationerne**: I en evt. IT-løsning til unionen vil der være mulighed for at finde informationer om kampe, spillere, klubber, trænere samt dommere. Dommerne skal kunne registrere hændelser der sker i en kamp – men andet udover skal gå igennem turneringsledelsen fx ændringer i kampe og andre oplysninger.
  + **Informationer er konsistent og samlet et sted**: Med pen og papir er man nød til at have en kopi for hver person, der skal se oplysningerne, hvilket giver mulighed for at ikke alle kopier har de samme informationer. Med et IT-system vil alle informationerne være samlet et sted og man kan få adgang til dem fra en computer.
  + **Sparer lagerplads**: Når unionen går fra papir og pen, vil de mærke en markant ændring i pladsmængden. Nu skal de ikke bekymre sig om lagerplads eller kontor plads til at opbevarer flere års papirer om diverse kampe, samt papirer på klubber, spillere, trænere osv.

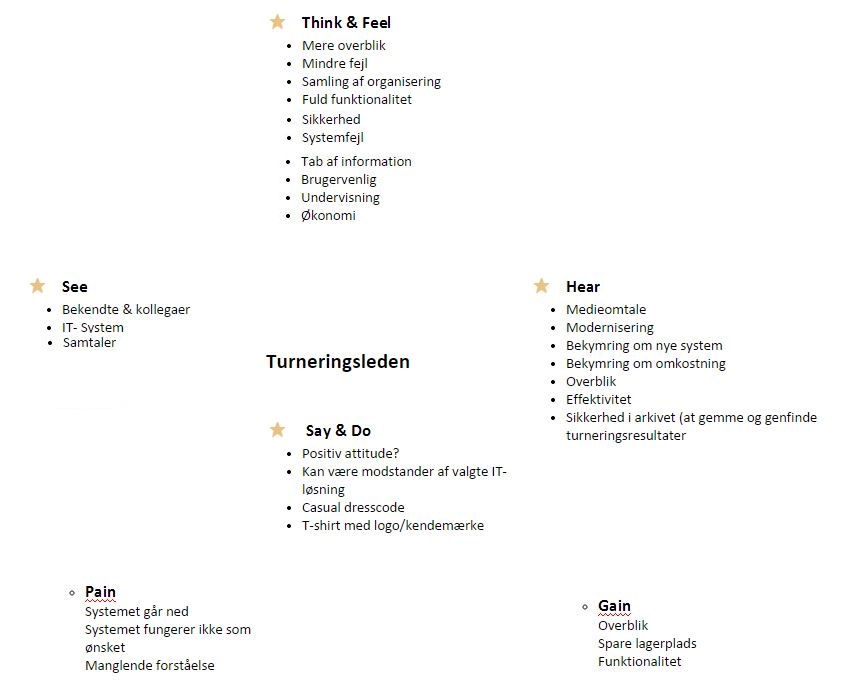
### Ulemper ved et digitalt system:

* **Yderligere elektronisk udstyr:** Når vi skal have implementere et program, kræver det at man skal have det nødvendige udstyr – og fordi at vi her både skal have lagerplads til at kunne gemme de oplysninger der oprettes og vi skal have kommunikation imellem pc'erne der benytter sig af dette program. Det er ikke et krav at programmet skal på flere pc'er, men hvis dette er løsningen eller ønsket, skal vi have implementeret muligheden for kommunikation.
* **System vedligeholdelse:** Når vi arbejder eller benytter programmer eller systemer, er det vigtigt at disse bliver holdt opdateret. Dette kan være en ulempe da der er varierende EDB forståelse i turneringsledelsen som skal benytte sig af dette program og derfor kan det kræve, at der fx bliver ansat en ekstra der står for vedligeholdelse.
* **Systemfejl:** Disse kan opstå ved en simple fejl indtastning eller kommunikation i IT-løsningen. Der er også risiko for at serveren går ned eller at informationen går tabt under overførsel.
* **Sikkerheden:** Ved alle systemer er der sikkerhedsrisici. Der er både tale om misbrug af informationer samt hacking – og man kan kun sikre sig til en hvis grad.
* ”**Kend dit system”:** Når man går fra pen og papir til et system skal man omstille sig mentalt. Vi vil derudfra gerne kunne eliminere evt. spørgsmål eller tvivl ved denne omstilling ved at lave et brugervenligt program og eventuelt undervise i det færdige program, samt kunne udgive nogle manualer til evt. pludselige opstået tvivlspørgsmål til brug af programmet.
* **Økonomi:** Ved implementering af IT-løsningen kan der forekomme nogle økonomiske overvejelser – både før og efter implementeringen, da IT-løsningen er et system til unionen og der er en udgift i at få løsningen lavet.

## Personlighedstræk hos bruger

Vi har valgt at benytte os af modellen ”The Empathy Map” da vi her kan få et analyserende indblik i hvordan systembruger/køber tænker ved en eventuel implementering af IT-løsning.

”The Empathy Map” er en model bygget på en analyserende forståelse af en eventuel systembruger/køber og deres reaktion på implementeringen af en IT-løsning. Da vi ikke kender personen eller deres reaktion, er vi nød til at kunne danne os et billede af hvordan denne person er som person og hvordan deres hverdag fungerer. Dette gør vi ved at kigge på de 6 forskellige segmenter i modellen:

* **Think & Feel** – Hvad tænker systembruger/køberen om IT-løsningen? Hvad er systembruger/køberens bekymringer og hvilke værdier har systembruger/køberen?
* **See** – Hvordan ser systembruger/køberens miljø ud? Hvilken slags venner har systembruger/køberen?
* **Hear** – Hvad siger systembruger/køberens venner? Hvad siger chefen? Hvilke indflydelser kan der være?
* **Say & Do** – Hvad er systembruger/køberens attitude i offentligheden? Hvordan er systembruger/køberens opførsel imod andre?
* **Pain** – Hvad er systembruger/køberens frygt, frustrationer og forhindringer?
* **Gain** – Hvad vil systembruger/køberen have/behov for? Hvad får systembruger/køberen ud af IT-løsningen? 

Vores systembruger er et medlem af turneringsledelsen og delvis en dommer, hvis der er ønske om decentralisering i organisationen. Dog har vi valgt at tage en person fra turneringsledelsen, som skal benytte sig af denne IT-løsning, da det er den person samt stillingen, der skal benytte sig af programmet mest.

Vores person fra turneringsledelsen, som vi vil lave en ”Empathy map” over, er en mand – ved navn Brian. Brian er gift, har børn og er i 30’erne. Han starter sin dag derhjemme, hvor han vågner op og han skal gøre sig selv og børnene klar til dagen. Han skal ikke skynde sig med nogle ting om morgen, men han skal i sidste ende få kørt ungerne i skole/børnehave.

Brian ligger vægt på at han kan have en stille og rolig morgen uden stress og jag, så han og familien spiser morgenmad sammen og der er et klart overblik over tingene.

Undervejs til hans arbejde hører han nyheder og den indskudte musik der nu kommer i radioen.

Når han møder på arbejdet møder han sine kollegaer, og der blive udvekslet en frisk godmorgen mellem dem – og derefter henter han sin morgenkaffe hvor snakken går yderligere mellem sine kollegaer.

Brian og hans kollegaer er ikke formelt klædt på, de går for det meste i træningstøj når de er i klubhuset, men når de er ude til kampene for at observere og notere har de en anden dresscode. Det de typisk kommer i er en T-shirt med deres logo på og et par jeans, og eventuelt en trøje til hverdag.

Undervejs til hans kontor vil han komme til høre for de forskellige holdninger der vil komme af IT-løsningen, og der er delte meninger. Han hører om bekymringer i forhold til omkostningerne af at købe systemet, om kan alt det de er blevet lovet og om det er nemt at bruge. Brian mener selv at lige netop en IT-løsning vil hjælpe på så mange forhindringer der ligger i den nuværende arbejdsgang. Han tænker for sig selv at det netop vil være nemmere at få overblik og da der ikke skal laves flere papirudgaver, der skal uddeles, kan der bruges mere tid på fodbold i stedet for det administrative.

Men ligeså har han nogle bekymringer han døjer med. Hvor sikkert er systemet? Man hører jo flere episoder med hackere, der har tvunget sig adgang til et system. Hvis der er fejl i systemet vil vi så miste information? Eller er der back ups af informationerne? Vil den være brugervenlig nok til at vi alle kan benytte os af IT-løsningen? Hvilken operativ system skal IT-løsningen køre på? Kan både deres Mac og Windows computere kører programmet.

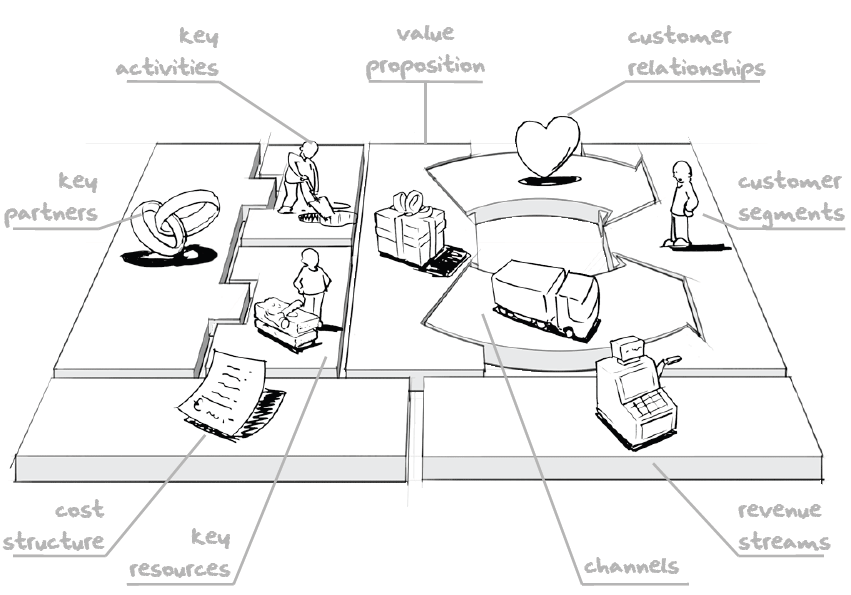
Da Brian er positiv omkring IT-løsningen holder han en positiv attitude over for hans omgivelser. Dog er nogle af hans kollegaer modstander af implementeringen, men de er i undertal i forhold til den mængde der stemte om at få implementeret en IT-løsning.

Brian som person ligger meget vægt på overblik og fuld funktionalitet – tingene skal kunne fungere uden problemer, men dette betyder ikke at han er konfliktsky, han tager gerne en konflikt op, men kun så længe at han selv kan se flere positive aspekter af problemet end negative.

## Udfordring af efterspurgte IT-løsning

Når man får sendt en ide ud i organisationen til en IT-løsning skal man overveje den nuværende forretningsmodel for organisationen og derfra prøve at udvikle forretningsmodellen til at den kommer til at fungere med IT-løsningen.

Vi har valgt at analysere og udfordre vores forslag til en IT-løsning ud fra ”**Buisness Model Canvas**” som består af 9 segmenter der hver spiller sammen med hinanden, og som skal give et billede af hvordan organisationen fungerer og om den kan forbedres yderligere.



De 9 segmenter i ”**Buisness Model Canvas**”:

1. Customer segment

* Kunderne er SBU selv, da IT-løsningen er beregnet til at gøre den administrative hverdag nemmere og give et større overblik over organisationens mål.

1. Value proposition

* IT-løsningen vil give organisationen overblik, en effektivitet da spildtid bliver formindsket og opgavefordelingen bliver nemmere gjort, med det vil der også komme en bedre funktionalitet.

1. Customer relationships

* Fordi organisationen vil have flere end en medarbejder af turneringsledelsen på IT- løsningen, er det vigtigt at den information, der er i systemet er troværdig.

1. Channels

* IT-løsningen kommer til at fungere til PC med operativsystem Windows. Mac kan diskuteres om det er nødvendigt.

1. Key activities

* De aktiviteter organisationen foretager sig er at arrangere kampe samt registrere resultater af afholdte kampe.

1. Key partners

* Partnere hos organisationen er tilmeldte klubber og DBU.

1. Key resources

* Organisationens ressourcer er information, da den bliver produceret af de kampe der afholdes og spilles af klubberne. Det er organisationens ansvar derefter at få informationen ud.

1. Cost structure

* Omkostninger for organisationen i forhold til IT-løsningen, vil være at få anskaffet sig hardwaren til løsningen. Softwaren (IT-løsningen) er ligeså en omkostning, da der skal løn til de programmører der har siddet og arbejdet på deres software.

1. Revenue streams

* Fordi at organisationen får en IT-løsning, vil det give dem en højere funktionalitet og effektivitet. Der vil formentlig blive nogle besparelser at hente i det lange løb. Da organisationens medarbejder er frivillige kan disse ikke skæres på for at opnå en besparelse.

Hvis vi så skal prøve og udfordre den nuværende forretningsmodel via. ”**Buisness Model Canvas**” kan vi kigge på den nuværende teknologi. Nu har vi mobiltelefoner der kan gå på nettet samt en nyere teknologi Tablets, som ligeledes kan gå på nettet. Hvis vi nu forestiller os at umiddelbart ¾ dele af turneringsledelsen ejer eller har adgang til en tablet – hvis vi videreudviklede IT-løsningen til at kunne bruges på en tablet og organisationen får den implementeret – hvad sker der så?

Jo, der ville ske nogle ændringer i vores ”**Buisness Model Canvas**”, således at omkostningerne ville forhøjes da virksomheden bag IT-løsningen skal bruge tid på at videreudvikle den nuværende IT-løsning så den er kompatibel til en anden platform. Der vil også opstå en større decentralisering, da der nu ikke kun er udvalgte pc’er men nu også tablets, der kan få adgang til IT-løsningen og organisationens oplysninger og informationer. Dette vil også hæve nogle spørgsmål om sikkerheds risici samt overvågning af systemet.

Dette har dog ikke kun negative men også positive effekter – flere mennesker kan sidde og styre dette, hvilke betyder at der ikke vil være kamp om hvem der skal benytte sig af hvad, og hvornår – samt vil der blive mindre arbejde pr. mand.

Når vi så har fokus på den produceret information i organisationen, skal vi så ikke overveje om eventuelle fans/offentligheden kan få adgang til udvalgt informationer via IT-løsningen? Da organisationen selv udgiver ligatallene elektronisk er dette spørgsmål oplagt, men dog skal vi tage højde for hvilken information vi giver offentligheden adgang til. Vi skal fx være forsigtig med at give dem oplysninger på den enkelte spiller, da vi har hans eller hendes CPR nummer knyttet til vedkommendes spillerprofil. Så sådanne ting skal udelades. De informationer vi kan give dem adgang til er klubberne – navnene på dem samt eventuelt hvor de har adresse, kommende kampe – så man eventuelt kan komme og se der bliver spillet kamp. Dette kan også være med til at skabe en større fan base for de enkelte klubber ved at vi giver offentligheden en adgang til udvalgt information.

Den IT-løsning vi har valgt, er valgt ud fra vores mening af hvad organisationen har behov for og hvad vi kan udvikle/levere til dem. Systemet er samtidig lavet så man nemt ville kunne skifte brugergrænseflade, hvis der er tale om at udvikle til en app eller fornyelse.

# Systemudvikling

I systemudvikling er vi blevet bedt om at udarbejde en analyse, der skal resultere i en domænemodel, der beskrives ved et UML klassediagram med klasser, attributter og strukturforbindelser i form af associationer med multipliciteter og kardinaliteter – samt eventuelle generaliseringsstrukturer.

Dermed skal vi også udarbejde en analyse af anvendelsesområdet, der skal resultere i beskrivelser af systemets samlede brugsmønstre i et brugsmønstrediagram. Dette skal vise aktører og brugsmønstre og forbindelser imellem disse.

Vi skal også have en mere detaljeret tekstlig beskrivelse af de viste brugsmønstre, i form af successcenarier, der viser normalforløbet og et eller flere specialscenarier, der viser fejl- eller specialtilfælde. Derudover skal vi have en fuldt udbygget brugsmønsterbeskrivelse for de vigtigste brugsmønstre.

Derudover skal vi udarbejde et design af systemet, hvor vi skal realisere brugsmønstrene og det resulterer i en udbygning af domænemodellen til en egentlig designmodel, hvor niveaudelt arkitektur vises som komponenter med afhængigheder imellem, og hvor hver komponent er vist med klasser der beskrives med navn, attributter og metoder.

Desuden skal der udarbejdes en plan af gruppens arbejde, der viser, hvad der skal udføres hvornår og af hvem.

## Kravspecifikation fra systembruger/køber

Følgende kravspecifikationer er blevet stillet fra Sjællands Boldspil-Union, som skal indgå i deres bestilte system. Kravspecifikationen er blevet stillet op med to kategorier – mindstekrav og konkurrenceparametre.

Disse to kategorier kan også betegnes med to andre udtryk: “need to have” som repræsenterer mindstekravene og “nice to have” som repræsenterer konkurrenceparametrene.

Mindstekravene er udelukkende hvad kunden behøver i programmet, hvorimod konkurrenceparametrene er til at gøre programmet og kunden konkurrencedygtig – dvs. de har noget som, forhåbentlig, ingen andre har, eller i som er bedre end det som de andre har.

I mindstekravene går vi i dybden med hvad SBU ønsker og hvad de skal have i deres program med denne kravspecifikation. I “nice to have” følger vi op på evt. flere ønsker SBU har til udvikling af programmet – dette er dog usikkert om dette kan implementeres samt om deadline er sandsynlig at nå ved disse ekstra implementeringer.

### Mindstekrav:

1. Vi skal lave et informationssystem der kan håndtere registreringer samt planlægning.
2. Det skal være muligt i programmet at tilrettelægge samt registrere en spilleplan, som skal indeholde dato, tid og sted – her skal der ske en koordination så at alle holdene kommer til at spille på hjemmebane kun en gang - dog behøves alle hold ikke at være tilmeldt en spilleplan da der forekommer en spilleplan per division.
3. Turneringsledelsen skal have speciel adgang i programmet så de kan ændre tid og dato for en bestemt kamp. Dette vil være klubbens ansvar at angive hvis en kamp skal ændres.
4. Programmet skal kunne håndtere klubskifte.
5. Programmet skal kunne håndtere spillerskift.
6. Programmet skal kunne håndtere op- og nedrykning af division.
7. Programmet skal kunne registrerer oplysninger om kampens afvikling.
8. Dommere samt spillere skal kunne bedømmes for deres præstationer ved kampe.
9. Sæsoner skal kunne oprettes ved sæsonskift samt kunne gemmes i en historik.
10. Programmet skal kunne registrere oplysninger om:

* Hver enkelte spiller
* De enkelte kampe
* Kampens hændelser

### Konkurrenceparametre:

1. Login til turneringsledelsen (her prøver vi at beskytte os imod hacking eller misbrug af sikkerhed).
2. Systemet registrerer startopstilling for hvert hold i kampregistrering.
3. Systemet kan registrere udskiftninger i kamprapporten.
4. Topscorelister for divisionerne.
5. Lister over flest gule/røde kort for hold/spillere.
6. Registrering af dommere (hver dommer kun dømme én kamp per turneringsrunde).
7. Bedømmelse af dommere og/eller spillere for hver spillet kamp. Lister over bedst præsterende dommere og spillere i hver division.

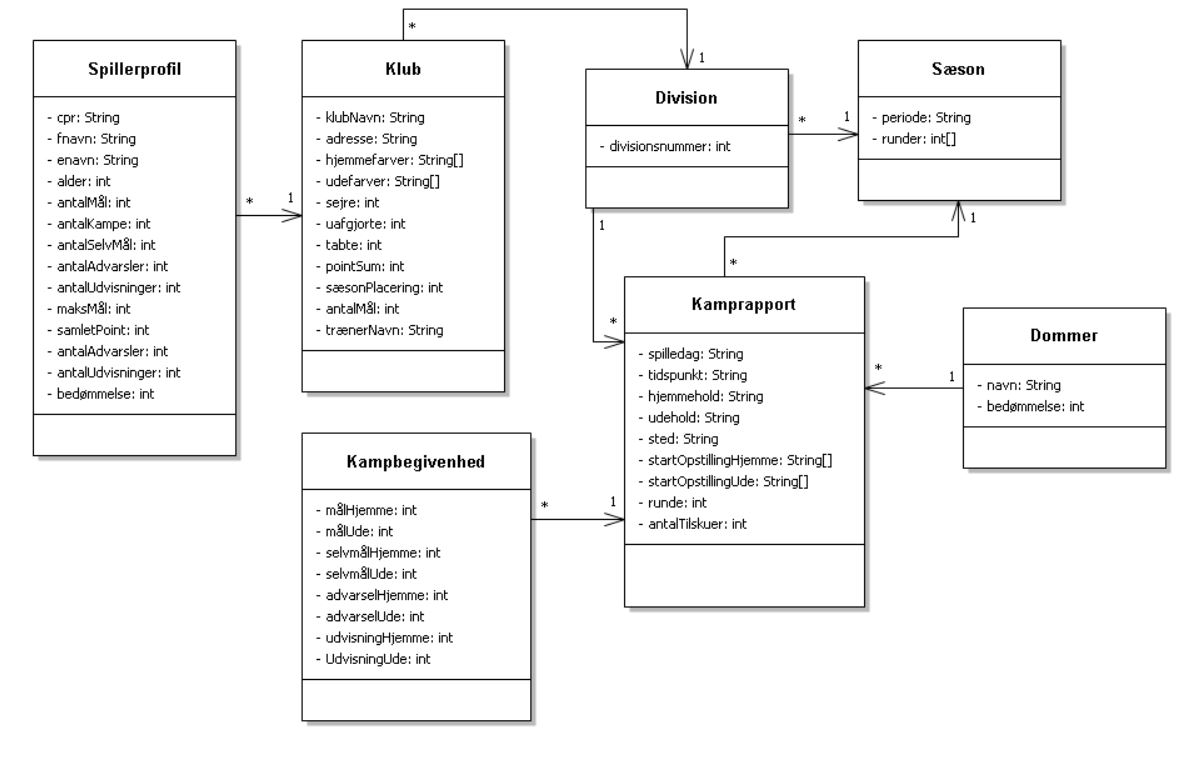
## Domænemodel

I dette afsnit vil vi udarbejde en analyse der resultere i en domænemodel. Dette gør vi ved at beskrive et UML klassediagram med klasser, attributter og strukturforbindelser i form af associationer med multipliciteter/kardinaliteter.

Ved observation af casen er vi kommet frem til følgende klasser:

* Spillerprofil.
* Klub.
* Division.
* Kampbegivenheder.
* Kamprapport.
* Sæson.
* Dommer.

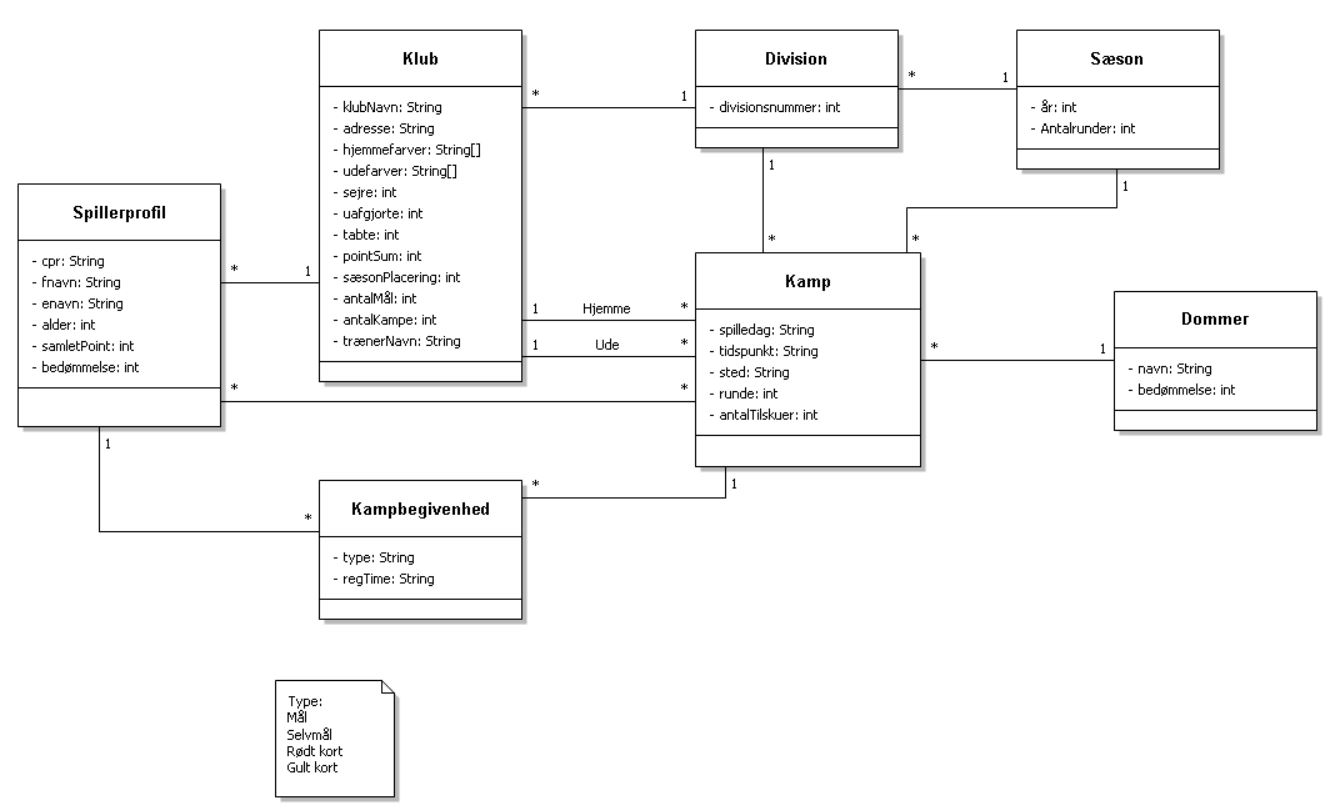
Derefter har vi kigget på hvad disse klasser skal have af attributter og hvordan de afhænger af hinanden.



Senere i forløbet i løsningen af opgaven, har vi opdaget nogle problemer ved den tidligere model. Der er blandt andet opstået redundans, hvilket betyder at vi har gentagende data igennem domænemodellen. Derfor har vi været nødt til at tage domænemodellen op for at gennemgå den igen.

De ændringer der er sket i modellen, er følgende;

* De akkumuleret data i klassen spillerprofil er blevet fjernet.
* I kampbegivenhed skulle vi have helt nye attributter – da vi før havde angivet begivenhederne som attributter, i stedet for type.
* I klassen Kamp er der blevet sat associationer imellem den og Klub – og imellem Kamp og Spillerprofil, samt at navnet er blevet ændret til Kamp i stedet for Kamprapport.
* I klassen Sæson er attributten periode omdøbt til år – samt at runder er blevet omdøbt til Antalrunder og ikke længere et *array*.

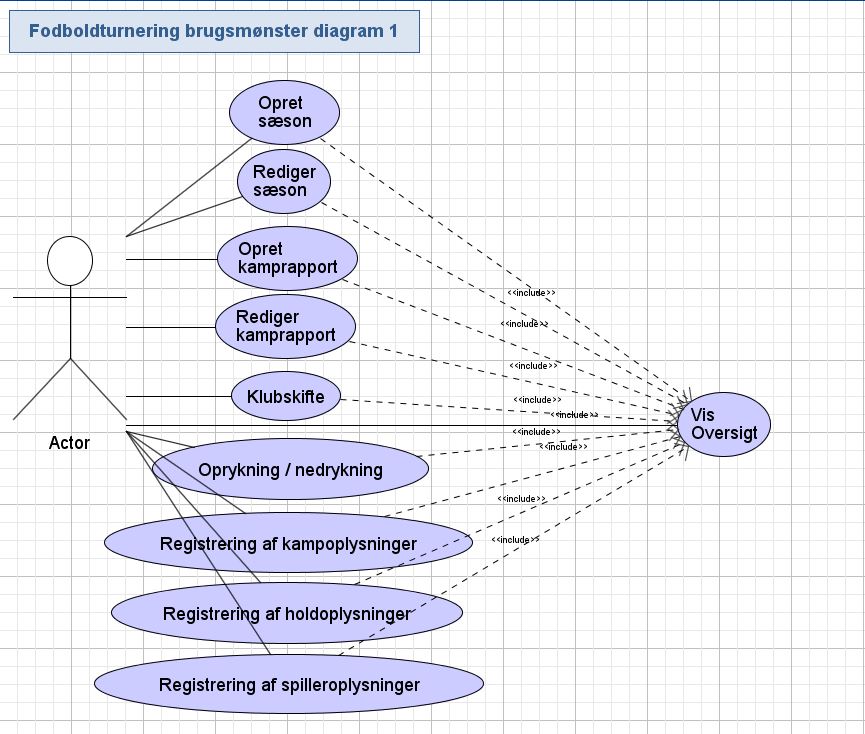


Grundet af manglende tid, og at fejlene er opdaget for sent, kan vi ikke nå og implementere denne model i programmet. Dette betyder så at vores færdige program er bygget ud fra det gamle klassediagram og script til database.

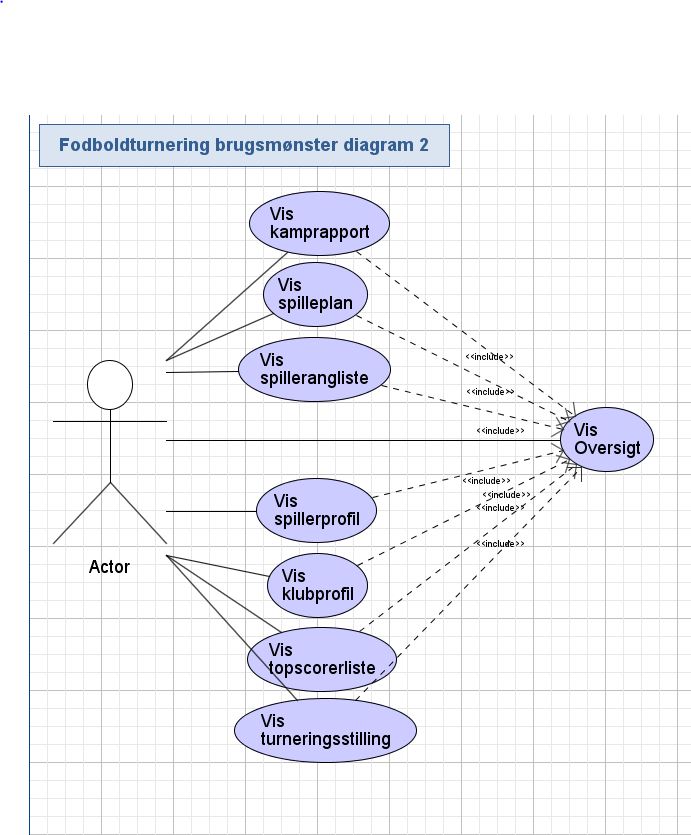
## Anvendelsesområder

I dette afsnit vil vi udarbejde en analyse af anvendelsesområdet, der resultere i en beskrivelse af systemets samlede brugsmønstre i et brugsmønstre diagram, der viser aktører og brugsmønstre imellem dem. Derudover vil vi beskrive et successcenarie, der viser normalforløbet og et eller flere specialscenarier der viser fejl eller specialtilfælde.

Nedestående diagrammer viser brugsmønstret for en aktør i SBU. Brugsmønsterdiagram 1 viser registering og oprettelse. Brugsmønsterdiagram 2 viser udtræk af systemet.



Brugsmønsterdiagram



Brugsmønsterdiagram

### Brugsmønster 1

**Navn**: Opret sæson

**Beskrivelse**: Oprettelse af en ny sæson med opdeling af divisioner og indsættelse af hold.

**Aktører**: Ledelse

**Hændelser**: Man skal være inde i programmet og der må ikke allerede være en aktiv sæson.

**Startkrav**: En sæson skal ikke være aktiv.

**Slutkrav**: En ny sæson er blevet oprettet med dertilhørende inddeling af divisioner og hold. Nye sæsoner kan ikke oprettes.

**Successcenarier**:

1. Brugeren trykker på ny sæson i brugergrænsefladen.
2. Brugeren indtaster de nødvendige oplysninger.
3. Brugeren opretter sæsonen.

**Specialscenarie**:

Forkert dato

1. Brugeren trykker spå ny sæson i brugergrænsefladen.
2. Brugeren indtaster de nødvendige oplysninger.
3. Indtastningen er ugyldig da datoen er ugyldig.
4. Indtast ny dato.

Allerede aktiv sæson

1. Brugeren trykker spå ny sæson i brugergrænsefladen.
2. En ny sæson kan ikke oprettes fordi en aktiv sæson allerede eksisterer.

Udvidelser: Opret kamprapport. Når man opretter en sæson kan man oprette alle de kampe der skal spilles i løbet af sæsonen.

### Brugsmønster 2

**Navn**: Opret kamprapport

**Beskrivelse**: Oprettelse af kamprapport med oplysninger om hold, tid og sted.

**Aktører**: Ledelse

**Hændelser**: Man skal ind i programmet og der skal være en aktiv sæson.

**Startkrav**: Der skal være en aktiv sæson.

**Slutkrav**: Der er blevet oprettet en ny kamprapport.

**Successcenarie**:

1. Brugeren opretter en ny kamprapport når kampens dato fastsættes.
2. Brugeren indtaster tid, sted og hold der skal spille.
3. Brugeren opretter kamprapporten som ligger klar når kampen spilles.

**Specialscenarie**:

Allerede en kamp

1. Brugeren opretter en ny kamprapport når kampens dato fastsættes.
2. Der er allerede en kamp med en af de hold i denne turneringsrunde
3. Ændrer runde eller annuller en af kampene.

Ugyldig indtastning

1. Brugeren opretter en ny kamprapport når kampens dato fastsættes.
2. En af oplysningerne er ugyldige.
3. Indtast de rigtige oplysninger.

**Udvidelser**: Registrer kampbegivenhed. Efter kampen er blevet oprettet kan der tilføjes oplysninger til kamprapporten i form af kampbegivenheder.

### Brugsmønster 3

**Navn**: Registrering af kampoplysninger

**Beskrivelse**: Tilføjelse af kampbegivenheder til kamprapporten i form af mål, selvmål, advarsler eller udvisninger.

**Aktører**: Ledelse eller dommer

**Hændelser**: Der skal være en kamprapport over en kamp der bliver spillet nu.

**Startkrav**: Der skal være en kamprapport.

**Slutkrav**: En kampbegivenhed er blevet tilføjet til kamprapporten.

**Successcenarie**:

Registrering af kampoplysninger

1. Brugeren går ind i den oprettede kamprapport.
2. Brugeren indtaster den eller de begivenheder der er sket.
3. Brugeren gemmer de nye oplysninger.

### Brugsmønster 4

**Navn**: Registrering af holdoplysninger.

**Beskrivelse**: Registrering af begivenheder, der påvirker holdet.

**Aktører**: Ledelse eller Dommer

**Hændelser**: Der skal være en kamprapport over en kamp der bliver spillet nu.

**Startkrav**: Der skal være en kamprapport.

**Slutkrav**: Oplysninger om holdet bliver opdateret.

**Successcenarie**:

Registrering af holdoplysninger

1. Når kamprapporten bliver inaktiv så bliver holdene opdateret.
2. Brugeren indtaster yderligere oplysninger hvis der er behov.

### Brugsmønster 5

**Navn**: Vis turneringsstilling.

**Beskrivelse**: Viser stillingen over holdene i en bestemt division.

**Aktører**: Ledelsen.

**Hændelse**: Man klikker ind på en bestemt division.

**Startkrav**: Divisionen skal være oprettet.

**Slutkrav**: En sorteret liste over holdene er blevet vist.

**Successcenarie**:

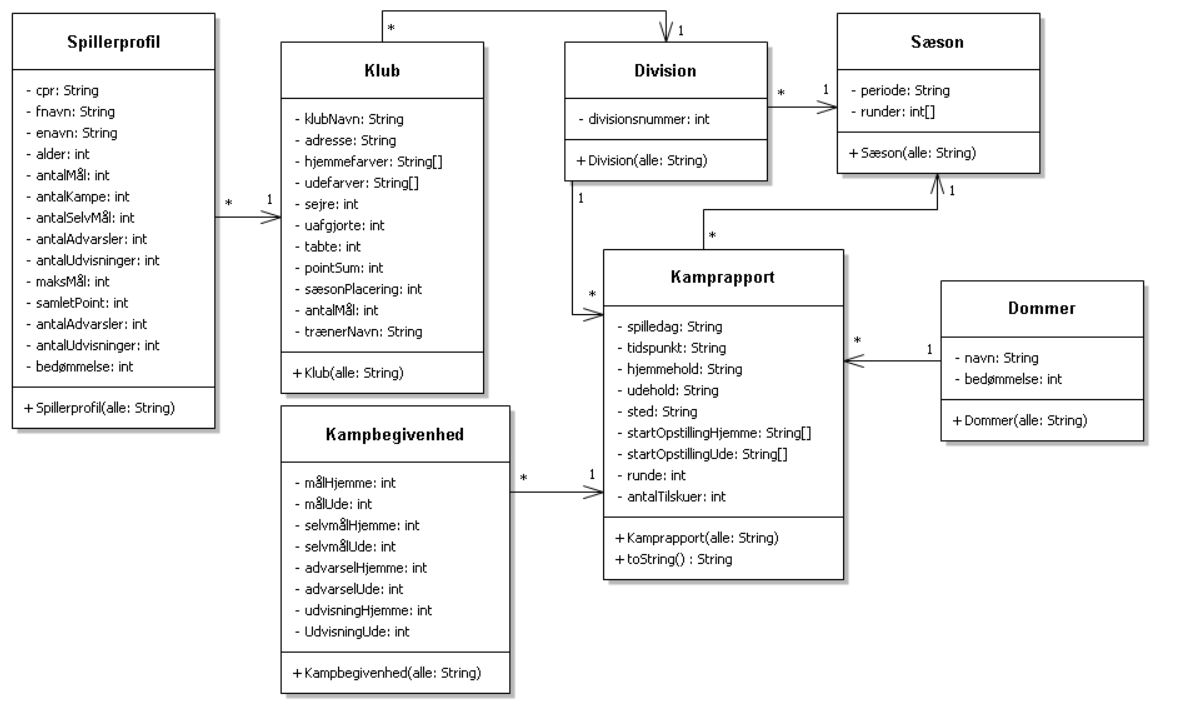
Vis turneringstilling

1. Brugeren trykker sig ind på den rigtige sæson.
2. Brugeren trykker sig ind på den rigtige division.
3. Divisionen er sorteret efter point.

## Designmodel

I dette afsnit skal vi udarbejde et design af systemet, hvor realiseringen af vores brugsmønstre resulterer i en udbygning af den nuværende domænemodel til en designmodel. Her vises niveaudelt arkitektur som komponenter med afhængigheder imellem, og hvor hver komponent er vist med klasser der beskrives med navn, attributter og metoder.

Nedestående kan vi se det færdige system, dog bygget videre fra den gamle domænemodel.



## Planlægning af projektet

Vi har planlagt at bruge ca. 1 uge pr. iteration. Derudover har vi løbende planlagt, hvad det er, som vi skulle lave de enkelte dage:

1. Iteration(overordnet analyse) ca. 1 uge:
2. UML klassediagram
3. Brugsmønstre
4. Klasser i klassediagrammet implementeres i programmet.
5. Virksomhedsanalyse (IT i Organisation)
6. Vurdere de organisatoriske fordele og ulemper ved det digitale system(IT i Organisation)
7. Rapport
8. Iteration(går i dybden) ca. 2 uger:
   1. Uddybe klasser I klassediagram
   2. Beskrive de vigtigste brugsmønstre.
   3. GUI implementeres i programmet.
   4. Database oprettes.
   5. Resterende del af IT i Organisation færdiggøres
   6. Rapport
9. Iteration(færddigørelse) ca. 1 uge:
   1. Beskrive og færdiggøre brugsmønstrene.
   2. Færdiggøre klassediagram.
   3. Implementere det resterende i programmet samt færdiggørelse.
   4. Rapport færdiggørelse.

Note: Observantrollen har været påtvunget, da arbejdet har foregået over Teamviewer, og derved har vi alle deltaget i arbejdet på flere niveauer.

7/11 – Fredag: Lav Use Case og klassediagram.

Lavet af: Andreas

*Observant: Jimmi & Stine*

8/11 – Lørdag: IT org (overfladisk).

Lavet af: Stine

*Observant: Andreas & Jimmi*

9/11 – Søndag: Uddybende klasse og Use Case over opret sæson (model klasser).

Lavet af: Andreas

*Observant: Jimmi & Stine*

10/11 – Mandag: Programmer opret sæson (model klasser).

Lavet af: Andreas

*Observant: Jimmi & Stine*

11/11 – Tirsdag: Virksomhedsanalyse, uddybe fordele/ulemper.

Lavet af: Stine

*Observant: Andreas & Jimmi*

12/11 – Onsdag: Påbegyndende rapport, Turneringsrunde klassen, kravspecifikation, vælge Use Cases til torsdag, uddybe opret sæson Use Case med scenarier og sekvensdiagram.

Lavet af: Andreas, Jimmi & Stine

*Observant:*

13/11 – Torsdag: uddybe Use Cases over funktioner der skal implementeres.

Lavet af: Andreas, Jimmi & Stine

*Observant:*

14/11 – Fredag: arbejde videre med virksomhedsanalyse og Use Case. Implementer nogle Use Case. Lavet af: Andreas & Stine

*Observant: Jimmi*

15/11 – Lørdag: Implementer vis turneringsstilling og registrer kampoplysninger, Evt. beskrivelse af brugsmønstre.

Lavet af: Andreas

*Observant: Jimmi & Stine*

16/11 – Søndag: fri.

Lavet af:

*Observant:*

17/11 – Mandag: Gå kontrolklasserne igennem. Lav database. Gå i gang med GUI (opret sæson).

Lavet af: Andreas, Jimmi & Stine

*Observant:*

18/11 – Tirsdag: Undervisning.

Lavet af:

*Observant: Andreas, Jimmi & Stine*

19/11 – Onsdag: Database – normalisering + data. Fortsæt med GUI

Lavet af: Andreas, Jimmi & Stine

*Observant:*

20/11 – Torsdag: Diskussion af IT org.

Lavet af: Andreas, Jimmi & Stine

*Observant:*

21/11 – Fredag: Lav GUI.

Lavet af: Andreas

*Observant: Jimmi & Stine*

22/11 – Lørdag: GUI + See, hear osv.

Lavet af: Andreas, Jimmi & Stine

*Observant:*

23/11 – Søndag: fri.

Lavet af:

*Observant:*

24/11 – Mandag: Påbegynd handler klasser.

Lavet af: Andreas

*Observant: Jimmi & Stine*

25/11 – Tirsdag: virksomhedsanalyse færdig.

Lavet af: Stine

*Observant: Andreas & Jimmi*

26/11 – Onsdag: Registrere begivenheder + lukke for registrering efter kamp + sortere efter hold + ændring af IP, port, DB navn, brugernavn og kode i DB + Udtræk turneringsstilling.

Lavet af: Andreas

*Observant: Jimmi & Stine*

27/11 – Torsdag: Design model, Brugsmønster og opdatere klassediagram.

Lavet af: Andreas & Stine

*Observant: Jimmi*

28/11 – Fredag: UML prog. + Udtræk kamprapport + Kamp rapport (skal kunne beregne score). Lavet af: Andreas

*Observant: Jimmi & Stine*

29/11 + 30/11 – Lørdag + Søndag: Skrivning på handler.

Lavet af: Andreas

*Observant:*

1/12 – Mandag: Rapport skrivning + retning af tidsplan + opret sæson.

Lavet af: Andreas, Jimmi & Stine

*Observant:*

2/12 – Tirsdag: Skrevet indledning af systemudvikling og programmering, samt lavet søgefunktion og færdiggjort opret sæson.

Lavet af: Andreas & Stine

*Observant:*

3/12 – Onsdag: Skrevet database afsnit i rapport, skrevet konklusion i IT i Organisation, påbegynd systemudviklings afsnit i rapport

Lavet af: Andreas og Stine

*Observant: Jimmi*

4/12 – Torsdag: Programbeskrivelse og exceptionhandling i programmering, systemudviklings afsnit videre skrives.

Lavet af: Andreas, Stine og Jimmi

*Observant:*

5/12 – Fredag: Programmeringsrapport.

Lavet af: Andreas

*Observant: Jimmi*

6/12 – Lørdag: Skrevet Programmeringsdelen færdig i rapport.

Lavet af: Stine & Andreas

*Observant:Jimmi*

7/12 – Søndag: Test og retning af rapport

Lavet af: Stine & Andreas

*Observant: Jimmi*

8/12 – Mandag: Aflevering af semester projekt

Lavet af: Stine, Andreas & Jimmi

*Observant:*

# Programmering

Vi skal lave et delsystem til holde styr på kampprogram og stilling i 3 divisioner, samt gemme kamprapporter for de spillede kampe, for en enkelt sæson. Programmet skal benytte den MySQL database I laver. Programmet skal være objektorienteret og skal oprettes med en GUI, og programmet skal være brugervenligt for den tilsigtede slutbruger, på grund af den varierende EDB forståelse i organisationen.

Hvert hold skal have tilknyttet et antal spillere. Vi skal dog ikke tage højde for holdskifte i løbet af sæsonen i dette program, en spiller er således kun tilknyttet et hold.

Programmet skal kunne oprette et kampprogram for en sæson for hver division. Hvert hold skal spille en hjemmekamp og en udekamp mod hvert af de andre hold.

For hver kamp (der ikke allerede er afsluttet) skal det være muligt at registrere følgende begivenheder:

* Scoring
* Selvmål
* Gult kort
* Rødt kort

For hver begivenhed skal man kunne registrere, hvor mange minutter inde i kampen begivenheder indtraf og hvilken spiller, der var involveret. Programmet skal sikre at man kun kan vælge spillere, hørende til et af de to hold i kampen, til begivenhederne.

Når kampen er færdig og alle begivenheder er registreret, skal man kunne registrere at kampen er afsluttet.

For afsluttede kampe, skal det være muligt at trække en kamprapport. Kamprapporten skal indeholde oplysninger om de 2 hold, der spillede med angivelse af hjemmehold samt en liste over alle begivenhederne i kampen i kronologisk rækkefølge.

Desuden skal kamprapporten ud fra begivenhederne beregne målscore.

Man skal kunne trække en turneringsstilling ud for hver division. Turneringsstillingen skal indeholde en oversigt over alle holdene i divisionen sorteret efter deres placering (stigende fra færrest point), desuden skal det fremgå, hvor mange kampe hvert hold har spillet, hvor mange point de har, hvor mange kampe de har vundet, spillet uafgjort og tabt samt holdets samlede målscore.

Det skal være muligt at ændre følgende oplysninger om databasen:

* IP/hostname
* Port
* Databasenavn
* Brugernavn
* Password

## Beskrivelse af programmet

Måden vi har grebet opgaven an på, er at vi først og fremmest har delt opgavens mængde op i iterationer - på denne måde har vi undgået forvirring i udviklings processen.

1. Iteration:

* UML-diagram af programmet laves.
* Programmer opret sæson (model klasser).

1. Iteration:
   * GUI implementeres i programmet.
   * Database oprettes.
   * UML-diagram af programmet færdiggøres.
2. Iteration:
   * Implementere det resterende i programmet samt færdiggørelse

Vi har et UML klassediagram, der viser hvordan programmet er bygget op. Programmet er bygget op i en MVC arkitektur. I modellaget har vi en sæson, der er opdelt i divisioner med et antal klubber, der hver har nogle spillere. I den sæson er der kampe, hvor der er tilknyttet en dommer pr. kamp. Bedømmelse og begivenheder har vi valgt at håndtere i disse klasser. Det har vist sig at begrænse vores muligheder for at registrere informationer omkring disse, men giver i stedet en akkumulerende funktion. Det sparer desuden tid, men er ikke relevant i et så lille system som vores.

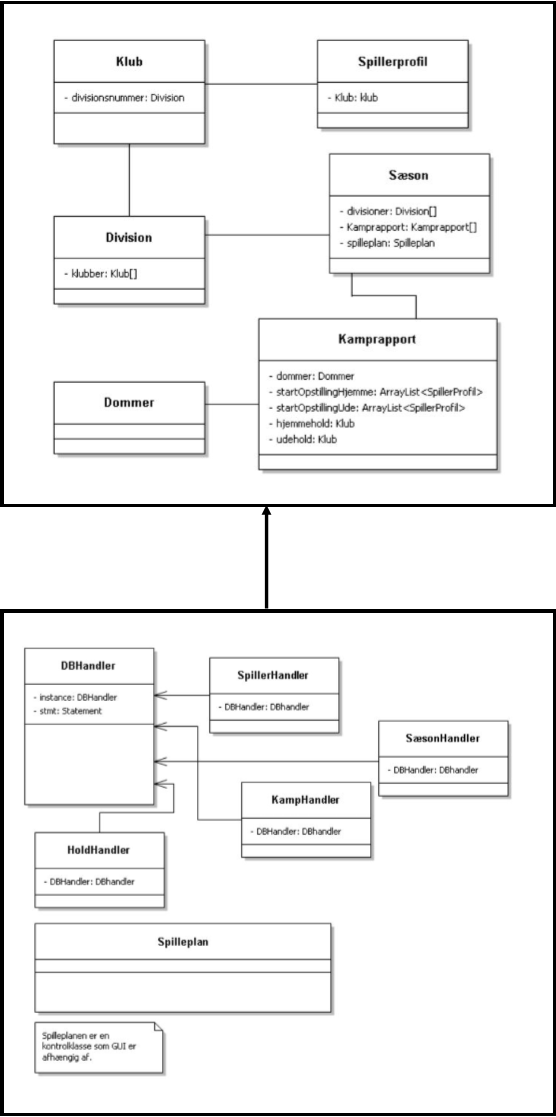
I kontrollaget har vi handlerklasser, der håndterer logikken mellem databasen og programmet. Vi har en databasehandler, der opretholder en forbindelse til databasen og så har vi handlerklasser til sæson, kamp, hold og spiller, hvor vi kan hente og manipulere informationer fra databasen. Hver af de klasser har metoder til at hente enten en eller flere entiteter fra databasen som enten returneres som et objekt eller en *ArrayList* af objekter.

Vi har ikke metoder til at oprette nye entiteter i databasen, men man kan godt oprette nye objekter i programmet, de vil bare blive glemt, når man lukker programmet. Det ville være ideelt, hvis de forskellige handlerklasser også kunne gemme ny information, men det har vi desværre ikke nået.

Vi har også en klase spilleplan, der opretter kampene. Den sikrer via metoden harSpillet(), at alle kampe er unikke.

Det er også i kontrollaget at vi har valgt at håndtere sortering af turneringshold. Det er gjort i holdhandleren ved at benytte os af algoritmen *insertion sort,* der er en sorterings algoritme, da den er en af de hurtigste sorteringsmetoder når vi har så få objekter, der skal sorteres. Pseudokode til *insertion sort* er fra bogen ”Introductin to Algorithms 3rd. Edition” s. 18.

I view-laget har vi en GUI klasse, der bruger logikken fra kontrollaget til at vise informationer i en brugergrænseflade.



### Forbedringer

I lyset af tiden der er gået, kan vi uden tvivl se en anden fremgangsmåde vi kunne have benyttet os af, så vi kunne have fanget fejlene i god tid, så vi kunne nå og implementere registrering af begivenhederne. Derudover ville vi gerne kunne implementere bedømmelse på samme måde, så vi kunne gemme informationerne i databasen.

Vi ville gerne have implementeret *insert* som en metode i programmets handlerklasser, for at kunne oprette nye data i databasen.

I programmet var vi valgt at implementere en søge funktion, da vi har med mennesker at gøre der har en varierende EDB forståelse – dvs. at vi skal give dem muligheden for at finde deres løsning/mål på en anden måde end den forventede.

## Exceptionhandling

Vi har valgt at minimere antallet af steder hvor *exceptions* kan forekomme. Det har vi for eksempel gjort ved at indsætte *Comboboxe*, så man kun kan vælge det som vi har bestemt. Ellers har vi *ClassNotFoundException* i *DBHandler* og *SQLException* i vores handlerklasser som er nødvendige i tilfælde at fejl i databasen. Vi har ikke givet mere specifikke fejlmeddelelser i dem, da vi ikke har implementeret metoder til at gemme i databasen, så al indtastning vil kun kunne ses i *GUI’en*.

## Test

Vi har desværre ikke lavet nogle test. Vi tænkte første på at teste programmet til sidst og vi kunne ikke finde ud af det på grund af manglende erfaring med at lave testcases.

# Database

I opgave formuleringen har vi fået af vide at vi skal lave et MySQL script som programmet skal benytte sig af og ved at programmet skal være objektorienteret skal det kunne oprette relevante objekter ud fra data i databasen.

Scriptet for oprettelse af database samt de indsatte værdier kan findes i bilag 1: Database.

## Konverterings algoritmen

1. Svage entiteter gøres stærke ved at der tilføjes en primær nøgle.

alter table spiller  
add primary key (cpr);

alter table klub  
add primary key (klubNavn);

alter table division  
add primary key (divisionsnummer);

alter table sæson  
add primary key (periode);

alter table dommer  
add primary key (navn);

alter table kamprapport  
add primary key (udehold, hjemmehold);

1. For hver stærk entitet oprettes en tabel, med registrering af primærnøglen.
2. For hver 1:1 forbindelse tilføjes primærnøglen fra den ene tabel som fremmednøgle i den anden.

Der er ingen 1:1 forbindelser i databasen

1. For hver 1:n forbindelse tilføjes primærnøglen fra 1-siden som fremmednøgle på n-siden.

alter table spiller  
 add foreign key (klubNavn) references klub;

alter table klub   
 add foreign key (divisionsnummer) references division;

alter table division  
 add foreign key (periode) references sæson;

alter table kamprapport  
add foreign key (divisionsnummer) references division,  
add foreign key (periode) references sæson,  
add foreign key (navn) references dommer;

1. For hver n:m forbindelse oprettes en ny tabel med primærnøglerne fra de to tabeller som sammensat primærnøgle og med de to primærnøgler som fremmednøgler.

**Vi har ingen n:m forbindelse I databasen.**

1. For hver flerværdi attribut oprettes en ny tabel med primærnøglen og flerværdi attributten om sammensat primærnøgle. Samtidig fjernes flerværdi attributten fra den oprindelige tabel.

**Vi har ikke nogle flerværdi attributter.**

1. Hvis der findes forbindelser hvori der indgår mere end to entiteter skal der oprettes en ny tabel med primærnøglerne fra de deltagende entiteter som sammensat primærnøgle.

**Den er opfyldt.**

## Normalisering

Database normalisering er en teknik der organisere data i en database. Normalisering er en systematisk opdeling af enheder i tabellerne for at eliminere data redundans og uønskede kommandoer som Insert, update og delete. Det er en flertrinnet proces der deler data ind i tabelform ved at fjerne duplikeret data fra relations databasen.

Normalisering er brugt til primært to formål:

* Eliminere redundans(ubrugeligt) data
* Sikre data afhængigheder giver mening dvs. data er gemt logisk.

### Første normalform

En tabel er på første normalform, hvis alle domænerne består af uendelige elementer, dvs. der ikke må forekomme sammensatte attributter eller flerværdi attributter.

I vores database har vi hverken sammensatte attributter eller flerværdis attributter. De sammen satte attributter har vi undgået ved at ikke har sat dem sammen i nogle af vores entiteter og flerværdis attributterne har vi ikke, da fx telefonnummer ikke indgår. Telefon ses typisk som en flerværdi attribut da vi typisk har flere end en telefon til rådighed i vores hverdag. Det sammen kan vi sige med navne. I vores database har vi valgt og skille navne af på den måde at de får en attribut hver:

drop table if exists spiller;

create table spiller

(

fnavn varchar(20),

enavn varchar(20)

);

## Anden normalform

På anden normalform må der ikke være delvis afhængighed af en kolonne på en primær nøgle. Det betyder at på en tabel der har en sammensat primær nøgle, hvor kolonne i tabellen der ikke er en del af primær nøglen må afhænge på hele den sammensatte primær nøgle for at den kan eksistere.

Hvis en af kolonnerne er afhængig af en del af den sammensatte nøgle er tabellen ikke i anden normalform.

I vore database har vi et sæt af sammensatte primær nøgler og disser er fuldt afhængige;

alter table kamprapport

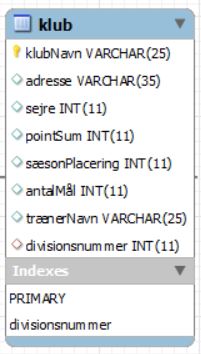
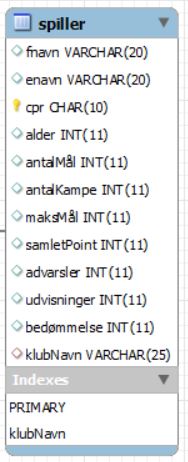
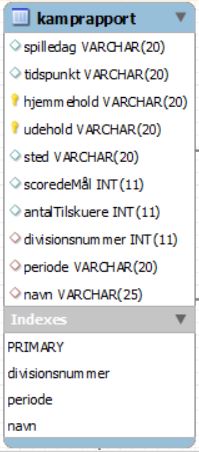
add primary key (udehold, hjemmehold);

### Tredje normalform

En relation R er på tredje normalform, hvis den er på anden normalform og det gælder, at der ingen non-prime attribut er transitiv afhængig af nogen primær nøgle i R.

Den transitive afhængighed kan forklares på nedestående formel:

I vores database har vi kigget på de store tabeller – dvs. de tabeller vi har vurderet til at kunne indeholde evt. transitive afhængigheder efter vi har kunne konkludere at vores tabeller er i anden normalform (se udklip af EER-diagram):



Efter at have kigget på de udvalgte tabeller har vi kunne konkludere at vores tabeller allerede var på tredje normalform – da vi ingen transitive afhængigheder har.

Ligeså har vi små tabeller i vores database, men har valgt ikke og tage dem i betragtning da vi mener de ikke har en transitiv afhængighed[[1]](#footnote-1).

# Bilag 1

## Database

I dette bilag vil i kunne finde de script til databasen er henholdsvis opretter og indsætter værdier i databasen. Koden er kopieret fra MySQL.

Oprettet database:

drop database if exists fodboldturnering;

create database fodboldturnering;

use fodboldturnering;

drop table if exists spiller;

create table spiller

(

fnavn varchar(20),

enavn varchar(20),

cpr char(10),

alder int,

antalMål int,

antalKampe int,

maksMål int,

samletPoint int,

antalSelvmål int,

antaladvarsler int,

udvisninger int,

bedømmelse int

)

Engine = InnoDB;

drop table if exists klub;

create table klub

(

klubNavn varchar(25),

adresse varchar(35),

sejre int,

uafgjorte int,

tabte int,

pointSum int,

antalKampe int,

sæsonPlacering int,

antalMål int,

trænerNavn varchar(25)

)

Engine = InnoDB;

drop table if exists division;

create table division

(

divisionsnummer int

)

Engine = InnoDB;

drop table if exists sæson;

create table sæson

(

periode varchar(20)

)

Engine = InnoDB;

drop table if exists kamprapport;

create table kamprapport

(

spilledag varchar(20),

tidspunkt varchar(20),

hjemmehold varchar(20),

udehold varchar(20),

sted varchar(20),

hjemmeMål int,

udeMål int,

selvmålHjemme int,

selvmålUde int,

advarslerHjemme int,

advarslerUde int,

runde int,

udvisningerUde int,

udvisningerHjemme int,

antalTilskuere int,

rundenummer int

)

Engine = InnoDB;

drop table if exists dommer;

create table dommer

(

navn varchar(25),

bedømmelse int

)

Engine = InnoDB;

Indsatte værdier i databasen:

use fodboldturnering;

delete from Spiller;

insert into Spiller

values

('0000000001', 'Peter', 'Møller', 0, 0, 0, 0 ,0 ,0, 0, 0, 'Aab');

delete from Klub;

insert into klub

values

('Aab', null, 0, 0, 0, 0, 0, 0, 0, 'Træner1',1),

('FC Midtjylland', null, 0, 0, 0, 0, 0, 0, 0, 'Træner2',1),

('OB', null, 0, 0, 0, 0, 0, 0, 0, 'Træner3',1),

('FC Nordsjælland', null, 0, 0, 0, 0, 0, 0, 0, 'Træner4',1),

('FC Vestsjælland', null, 0, 0, 0, 0, 0, 0, 0, 'Træner5',1),

('FC København', null, 0, 0, 0, 0, 0, 0, 0, 'Træner6',1),

('Hobro', null, 0, 0, 0, 0, 0, 0, 0, 'Træner7',1),

('Brøndby IF', null, 0, 0, 0, 0, 0, 0, 0, 'Træner8',1),

('Esbjerg FB', null, 0, 0, 0, 0, 0, 0, 0, 'Træner9',1),

('Randers FC', null, 0, 0, 0, 0, 0, 0, 0, 'Træner10',1),

('Silkeborg', null, 0, 0, 0, 0, 0, 0, 0, 'Træner11',1),

('Sønderjyske', null, 0, 0, 0, 0, 0, 0, 0, 'Træner12',1),

('AB', null, 0, 0, 0, 0, 0, 0, 0, 'Træner13',1),

('AC Horsens', null, 0, 0, 0, 0, 0, 0, 0, 'Træner14',1),

('AGF', null, 0, 0, 0, 0, 0, 0, 0, 'Træner15',1),

('Brønshøj', null, 0, 0, 0, 0, 0, 0, 0, 'Træner16',1),

('FC Roskilde', null, 0, 0, 0, 0, 0, 0, 0, 'Træner17',2),

('Fredericia', null, 0, 0, 0, 0, 0, 0, 0, 'Træner18',2),

('HB Køge', null, 0, 0, 0, 0, 0, 0, 0, 'Træner19',2),

('Lyngby', null, 0, 0, 0, 0, 0, 0, 0, 'Træner20',2),

('Skive', null, 0, 0, 0, 0, 0, 0, 0, 'Træner21',2),

('Vejle Boldklub', null, 0, 0, 0, 0, 0, 0, 0, 'Træner22',2),

('Vendsyssel FF', null, 0, 0, 0, 0, 0, 0, 0, 'Træner23',2),

('Viborg', null, 0, 0, 0, 0, 0, 0, 0, 'Træner24',2),

('Brabrand', null, 0, 0, 0, 0, 0, 0, 0, 'Træner25',2),

('FC Sydvest', null, 0, 0, 0, 0, 0, 0, 0, 'Træner26',2),

('Jammerbugt FC', null, 0, 0, 0, 0, 0, 0, 0, 'Træner27',2),

('Kjellerup', null, 0, 0, 0, 0, 0, 0, 0, 'Træner28',2),

('Kolding B', null, 0, 0, 0, 0, 0, 0, 0, 'Træner29',2),

('Kolding IF', null, 0, 0, 0, 0, 0, 0, 0, 'Træner30',2),

('Marienlyst', null, 0, 0, 0, 0, 0, 0, 0, 'Træner31',2),

('Middelfart', null, 0, 0, 0, 0, 0, 0, 0, 'Træner32',2),

('Næsby', null, 0, 0, 0, 0, 0, 0, 0, 'Træner33',3),

('Næstved', null, 0, 0, 0, 0, 0, 0, 0, 'Træner34',3),

('Obber IGF', null, 0, 0, 0, 0, 0, 0, 0, 'Træner35',3),

('Ringkøbing', null, 0, 0, 0, 0, 0, 0, 0, 'Træner36',3),

('Skovbakken', null, 0, 0, 0, 0, 0, 0, 0, 'Træner37',3),

('Svendborg', null, 0, 0, 0, 0, 0, 0, 0, 'Træner38',3),

('Thisted', null, 0, 0, 0, 0, 0, 0, 0, 'Træner39',3),

('Varde', null, 0, 0, 0, 0, 0, 0, 0, 'Træner40',3),

('Avedøre', null, 0, 0, 0, 0, 0, 0, 0, 'Træner41',3),

('Frem', null, 0, 0, 0, 0, 0, 0, 0, 'Træner42',3),

('Fremad Amager', null, 0, 0, 0, 0, 0, 0, 0, 'Træner43',3),

('Holbæk', null, 0, 0, 0, 0, 0, 0, 0, 'Træner44',3),

('Hvidovre', null, 0, 0, 0, 0, 0, 0, 0, 'Træner45',3),

('Nykøbing FC', null, 0, 0, 0, 0, 0, 0, 0, 'Træner46',3),

('Rishøj', null, 0, 0, 0, 0, 0, 0, 0, 'Træner47',3),

('Svebølle', null, 0, 0, 0, 0, 0, 0, 0, 'Træner48',3)

;

select \* from Spiller;

# Bilag 2

## Opgaveformulering

Formelle krav:

#### Projektet kører i perioden 6. november - 8. december 2014. Der er afsat tid til projektarbejde og vejledning i hvert af de fire fag indenfor denne periode sideløbende med almindelig undervisning. Det er op til de enkelte undervisere at prioritere tiden mellem undervisning og projektarbejde.

# 

Projektet skal løses i grupper på 3-4 personer. Det skal fremgå af rapporten, hvem der har ansvar for de enkelte afsnit. Hver studerende skal have hovedansvaret for mindst ét afsnit i to forskellige fag. Centrale afsnit i rapporten kan godt have flere ansvarlige.

Såfremt de studerende selv kan danne grupper af 3-4 personer, er der fri gruppedannelse. Hvis det ikke kan lade sige gøre at danne grupper selv, hjælper underviserne med at danne grupperne. Gruppedannelserne er bindende når de er godkendt af underviserne.

Specifikke krav til hvert fag står opført i opgavebeskrivelsen for det pågældende fag.

Opgaven skal afleveres i 4 eksemplarer på papir, samt 4 CD/DVD/USB med de dele, der skal afleveres elektronisk på kontoret, senest mandag d. 8. december 2014 kl. 12:00.

Opgaven skal forsynes med en forside, hvor det fremgår hvilken opgave det drejer sig om, samt hvilke personer, der indgår i gruppen. Forsiden skal underskrives af hvert medlem af gruppen.

Opgaven skal sidenummereres og der skal være en indholdsfortegnelse. Dokumentation, der ikke direkte indgår som en del af rapporten, skal samles i nummererede bilag i slutningen af rapporten. Bilagene skal fremgå af indholdsfortegnelsen.

God arbejdslyst!

Case beskrivelse:

Der skal udvikles et informationssystem til planlægning og registrering af en fodboldturnering.

Turneringen omfatter 16 hold fra 16 klubber fordelt på 3 divisioner. Forud for hver sæson tilrettelægges og registreres en spilleplan med de enkelte kampes spilledatoer og –tidspunkter samt spillesteder, idet holdene møder hinanden en gang på hjemmebane og en gang på udebane i hver sæson. Klubberne kan dog få ændret spilledato og –tidspunkt gennem turneringsledelsen.

Til hver klub er knyttet en træner og et antal spillere. Klubskifter finder almindeligvis sted mellem to sæsoner, men kan forekomme midt i en sæson.

Efter hver sæson er der op- og nedrykning af klubberne efter følgende regler. De 3 lavest placerede i 1. og 2. division rykker ned i henholdsvis 2. og 3. division, og de 4 lavest placerede i 3. division rykker helt ud af divisionsturneringen. Der sker en modsvarende oprykning.

I løbet af en spillesæson skal der ske en registrering af oplysninger om kampenes afvikling: Holdenes startopstilling, scorede mål, tilskuerantal med videre. Kampenes resultater opgøres ved, at der gives 3 point for vundne, 1 point for uafgjorte og 0 point for tabte kampe. Dommere og spillere bedømmes for deres præstationer, idet de tildeles 1 – 6 point efter en bedømmelsesskala med 1 som laveste bedømmelse.

Efter hver sæson skal der foretages en større justering af de registrerede oplysninger. Alle sæsonens resultater gemmes som historik, og der gøres klar til en ny sæson med op- og nedrykninger, spillerskifte o.s.v.

Systemet skal give mulighed for at registrere oplysninger om de enkelte spillere, de enkelte kampe og kampenes hændelser. Ud fra disse oplysninger skal der kunne dannes følgende udtræk:

**Turneringsstilling:**

Den aktuelle turneringsstilling for hver division samt de enkelte kampresultater for seneste spillerunde. Klubberne rangeres efter

1. Pointsum.

2. Målforskel.

3. Højeste antal scorede mål.

Informationsindhold: Division, spilledato, klubnavn, resultat, antal kampe, point, målscore.

**Kamprapport:**

Kampresultat og andre oplysninger om kampenes afvikling.

Informationsindhold: Division, spilledato, hjemmehold, udehold, scorede mål, scoringstidspunkt, tilskuerantal, advarsler, spillerudskiftning, dommer – og spillerpræstationer o.s.v.

**Spilleplan:**

Tidspunkter med videre for hver division for næste spillerunde.

Informationsindhold: Division, hjemmeholdets navn, udeholdets navn, spilledato, spilletidspunkt, spillested.

**Spillerrangliste:**

Opgøres for hver division, for seneste spillerunde samt for sæsonen til dato.

Informationsindhold: Division, Dato, Spillernavn, klubnavn, antal point.

**Spillerprofil:**

Spillerne rangeres efter antallet af scorede mål.

Informationsindhold: Spillernavn, klubnavn, antal kampe, antal mål, point på ranglisten, antal advarsler, antal udvisninger, alder, tidligere klubber, o.s.v.

**Topscorerliste:**

De 10 mest scorende spillere fra hver division.

Informationsindhold: Spillernavn, klubnavn, antal mål, antal kampe, højeste antal mål i en kamp.

**Klubprofil:**

Informationsindhold: Klubnavn, Adresse, farver, Antal turneringssejre, Placering de sidste 5 sæsoner, spillernavne, alder, træners navn, tidligere klubber.

**Opgaveformulering**

**IT i organisationen**

Sjællands Boldspil-Union (SBU) er en over 100 år gammel fællesunion for fodboldklubber på Sjælland. SBU varetager al administration af seriefodbold på Sjælland, men er organisatorisk underlagt Dansk Boldspil-Union (DBU) og dermed også Danmarks Idrætsforbund (DIF).

Med over 100 år på bagen er SBU en stolt lokalunion, som på trods af at være drevet af frivillige kræfter, har et tæt sammenhold og en ”SBU way of doing things”. Igennem alle år er stort set al turneringsadministration derfor sket ved hjælp af papir og pen, og kun over de seneste 15 år er ligatabeller blevet udgivet elektronisk på internettet. Af samme årsag har det været en stor beslutning, at anskaffe et automatiseret og digitaliseret system.

For at systemet bedst muligt tilpasses SBUs krav, skal I derfor:

* Udarbejde en organisatorisk beskrivelse for SBU, herunder turneringsledelsen
* Overveje hvor meget decentralisering af beslutningsrettigheder systemet kan medføre – kan dommere nu være behjælpelige med eksempelvis at flytte kampe og håndtere klager over rødt kort, eller skal ting som disse stadig ske hos turneringsledelsen?
* Vurdere de organisatoriske fordele og ulemper ved det digitale system
* Analysere personlighedstræk for systemkøber og/eller systembruger
* Udfordre den efterspurgte it-løsning og forretningsmodel – hvad hvis forretningsmodellen blev grebet anderledes an?

Husk på, at det afgørende for jeres opgavebesvarelser er den bagvedliggende argumentation fremfor det endelige produkt, der i dette fag er essentielt. Det er derfor ikke interessant at lave eksempelvis et Business Model Canvas, men argumentere **hvorfor**. Disse analyser bæres udelukkende af de argumenter og antagelser, der fremføres i opgavebesvarelsen.

**NB!** Alle de steder, hvor I foretager antagelser om enten SBU, deres ansatte, it-systemet eller noget helt fjerde, skal dette tydeligt ekspliciteres.

**Systemudvikling**

Ved tvivlsspørgsmål angående forståelsen af foregående tekst skal egne forudsætninger tydeligt fremgå af besvarelsen.

Udarbejd en analyse, der resulterer i en domænemodel, der beskrives ved

Et UML klassediagram med klasser, attributter og strukturforbindelser i form af

Associationer med multipliciteter / kardinaliteter

Og eventuelle generaliseringsstrukturer.

Udarbejd en analyse af anvendelsesområdet, der resulterer i beskrivelser af

Systemets samlede brugsmønstre i et brugsmønsterdiagram,

Der viser aktører og brugsmønstre og forbindelser imellem disse..

Mere detaljeret tekstlig beskrivelse af de viste brugsmønstre i form af

Et successcenarie, der viser normalforløbet og

Et eller flere specialscenarier, der viser fejl- eller specialtilfælde

En fuldt udbygget brugsmønsterbeskrivelse for de vigtigste brugsmønstre.

Eventuelt vist med System sekvens diagrammer.

Udarbejd et design af systemet, hvor realiseringen af brugsmønstrene resulterer i en udbygning af domænemodellen til en egentlig designmodel, hvor en niveaudelt arkitektur vises som komponenter med afhængigheder / dependencies imellem, og hvor hver komponent er vist med klasser, der beskrives med navn, attributter og metoder, gerne med parametre.

Udarbejd en plan for gruppens arbejde, der viser, hvad der skal udføres hvornår og af hvem. Det betyder, at der skal foreligge en aftalt arbejdsdeling i gruppen. Planen må gerne afspejle at udviklingsarbejdet er opdelt i iterationer, med indlagte målepunkter for projektets status efter hver afsluttet iteration.

Husk at knytte kommentarer til alle diagrammer / beskrivelser, der tydeliggør eventuelle overvejelser og valg, som I har foretaget under udarbejdelsen. Her kan eventuelle forudsætninger også beskrives.

**Programmering**

I skal lave et delsystem til at holde styr på kampprogram og stilling i tre divisioner, samt gemme kamprapporter for de spillede kampe, for en enkelt sæson. Programmet skal benytte den MySQL database I laver. Programmet skal være objektorienteret, dvs. at I skal oprette relevante objekter ud fra data i DB og arbejde med objekterne i programmet. Programmet skal oprettes med en GUI, og programmet skal være brugervenligt for den tilsigtede slutbruger.

Hvert hold skal have tilknyttet et antal spillere. I skal ikke tage højde for holdskifte i løbet af sæsonen i dette program, en spiller er således kun tilknyttet et hold.

Programmet skal kunne oprette et kampprogram for en sæson for hver division. Hvert hold skal spille en hjemmekamp og en udekamp mod hvert af de andre hold. Det er tilstrækkeligt at I benytter turneringsrunder svarende til en uges varighed i jeres kampprogram (*det er simplere end at arbejde med datoer*), således at hvert hold kun kan spille én kamp per turneringsrunde (det er dog tilladt at have runder, hvor nogle hold ikke spiller kamp, hvis gør det lettere at få kampprogrammet til at gå op). Der bør dog højst være 40 turneringsrunder på en sæson (*hvert hold skal spille 30 kampe*).

For hver kamp (der ikke allerede er afsluttet) skal det være muligt at registrere følgende begivenheder:

* Scoring
* Selvmål
* Gult kort
* Rødt kort

For hver begivenhed skal man kunne registrere, hvor mange minutter inde i kampen begivenheder indtraf og hvilken spiller, der var involveret. Programmet skal sikre at man kun kan vælge spillere, hørende til et af de to hold i kampen, til begivenhederne. Programmet skal ikke kontrollere at det er gyldige begivenheder (f.eks. to røde kort til en spiller i samme kamp, eller scoring af en spiller efter han har fået rødt kort).

Når kampen er færdig og alle begivenheder er registreret, skal man kunne registrere at kampen er afsluttet.

For afsluttede kampe, skal det være muligt at trække en kamprapport. Kamprapporten skal indeholde oplysninger om hvilke to hold, der spillede med angivelse af hjemmehold samt en liste over alle begivenhederne i kampen i kronologisk rækkefølge. Desuden skal kamprapporten ud fra begivenhederne beregne målscore.

Man skal kunne trække en turneringsstilling ud for hver division. Turneringsstillingen skal indeholde en oversigt over alle holdene i divisionen sorteret efter deres placering (flest point øverst), desuden skal det fremgå, hvor mange kampe hvert hold har spillet, hvor mange point de har, hvor mange kampe de har vundet, spillet uafgjort og tabt samt holdets samlede målscore.

Man behøver ikke kunne oprette hold og spillere i programmet. Det er tilstrækkeligt at I opretter disse direkte i DB i denne opgave.

Det skal være muligt at ændre følgende oplysninger om databasen:

* IP/hostname
* Port
* Databasenavn
* Brugernavn
* Password

Det kan f.eks. stå i en tekst-fil programmet indlæser eller sættes i nogle jTextFields i GUI’en, blot man har mulighed for at ændre på værdierne, så man kan køre programmet på maskiner, hvor MySQL f.eks. kører på en anden port eller har andet rootpassword.

I skal levere programmet sammen med et SQL script, der opretter databasen og indsætter data, således at man kan bedømme programmets funktionalitet. Der skal være mindst 16 forskellige hold i hhv. 1, og 2. division, hvert hold skal have minimum 11 spillere med forskellige navne (det er ok med enkelte gengangere, men det duer ikke med de samme 11 navne på hvert hold).

Hvis I har tid og lyst kan I implementere et eller flere af følgende ønsker (men disse ønsker er valgfri):

* Systemet registrerer startopstilling for hvert hold i kampregistrering.
* Systemet kan registrere udskiftninger i kamprapporten.
* Topscorelister for divisionerne
* Lister over flest gule/røde kort for hold/spillere
* Registrering af dommere (da der ikke er dato og tidspunkt på kampene, bør hver dommer kun dømme én kamp per turneringsrunde).
* Bedømmelse af dommere og/eller spillere for hver spillet kamp. Lister over bedst præsterende dommere og spillere i hver division.

### Beskrivelse af program

Skriv hvordan I har grebet opgaven an, hvad I har lagt vægt på, og hvilke valg har I foretaget.

Beskriv hvilke klasser I har valgt at dele programmet op i. Beskriv kort klassernes funktion i programmet og forklar hvad centrale metoder gør. Lav desuden et UML-diagram over de klasser I har valgt at lave, det skal fremgå hvilke afhængigheder, der er mellem klasserne.

### Exceptionhandling

Beskriv hvordan I har valgt at lave exception handling og begrund jeres valg. (*I skal hovedsagligt beskrive hvor I har beskyttet jer mod exceptions, samt om I håndterer exceptions, der hvor de opstår eller om I sender dem videre i systemet*.)

Tænk på hvem der skal bruge systemet og læse evt. fejlbeskeder, og sørg for at fejlbeskederne hjælper brugeren.

Sørg for at medtage hver fejlbesked fra jeres system i afsnittet og beskriv dem enkeltvis.

### 

### Test

Lav en funktionstest af disse funktioner i programmet:

* Oprettelse af kampprogram
* Registrering af begivenheder på en ikke afsluttet kamp
* Dannelse af turneringsstilling

Man kan med fordel dele testen af hver funktionalitet op i flere små test-cases, der hver især tester et lille specikt krav.

Hvert medlem i gruppen skal som minimum lave én test-cases.

Beskriv nøjagtig hvad I gør i hver test-case (*det skal være muligt at genskabe testen ud fra jeres beskrivelse*), samt hvad det forventede resultat og det faktiske resultat er. (*Find ud af hvad I forventer som resultat INDEN I tester, ellers er det nemt at overse et forkert resultat.*)

Hvis I finder fejl, som I ikke kan nå at rette, beskriv da, hvordan de evt. kunne rettes.

Skriv en konklusion på testen. (*Kan programmet tages i drift med de fejl, der evt. er fundet? Kan det afprøves af kunden?*)

1. http://www.studytonight.com/dbms/database-normalization.php [↑](#footnote-ref-1)