Competitie en Toernooisysteem

Applicatie voor wedstrijdorganisatie

Datum: 31 mei 2010

Afstudeerder: Paul Visser (1515194)
Bedrijfsbegeleider: Michel Everaert
1ste Examinator: Alex Jongman

Voorwoord

Deze scriptie heb ik geschreven voor mijn afstudeeropdracht bij de Nederlandse Volleybal Bond. Zelf ben ik een fanatiek (beach)volleyballer en twijfelde er dan ook geen moment aan dat er bij de Nevobo niets te doen zou zijn. Hoewel de communicatie in het begin wat stroef verliep zijn we uiteindelijk op een hele leuke opdracht, het automatiseren van toernooien en competities, gekomen. Deze opdracht gaf mij de kans om en alles over volleybaltoernooien te weten te komen en om een leuk systeem te maken waar behoorlijk wat uitdagingen in zouden zitten. Kortom, ik zou een hele leuke tijd tegemoet gaan!

Deze scriptie zou niet tot stand zijn gekomen zonder de vele hulp die ik heb gehad gedurende het traject. Mijn dank gaat vooral uit naar mijn docentbegeleider, Alex Jongman. Hoewel de echte begeleiding wat laat begon heb je me wel doen realiseren voor wie deze scriptie is en hoe ik sommige dingen inhoudelijk moet formuleren. Daarnaast gaat mijn dank uit naar een medestudent, Tim de Jager. Ook jij hebt mij inhoudelijk kunnen steunen en daarnaast heb je me ook enorm geholpen met spelfouten en het verwoorden van bepaalde zaken. Ook gaat mijn dank uit naar mijn vader en moeder. Jullie hebben mijn scriptie vooral naar een wat zakelijker niveau kunnen tillen. Tenslotte wil ik mijn tante Sytske uit Canada bedanken. Dat jij mijn scriptie tijdens je vakantie hebt willen lezen en daar opmerkingen op wou maken heeft mij een enorm plezier gedaan!

Paul Visser

Managementsamenvatting

Deze opdracht betreft het automatiseren van de wedstrijdorganisatie binnen toernooien en competities voor Beachvolleybal.

Beachvolleybal is een relatief jonge sport die nog volop in ontwikkeling is. Door het hele land worden steeds meer accommodaties opgezet en ook het aantal toernooien en deelnemers blijft stijgen. Ondanks deze groei worden er nog steeds ouderwetse methodes omtrent de organisatie gebruikt.

Om de groei ook op technisch gebied te ondersteunen wil de Nederlandse Volleybal Bond (Nevobo) een applicatie die de organisatie van toernooien en competities ondersteund. De belangrijkste eigenschappen van deze applicatie omvatten:

- Ondersteuning bieden bij het maken van poule indelingen en wedstrijdschema's;
- het verwerken van uitslagen en standen;
- communiceren met een bestaand inschrijfsysteem voor het ophalen van teams en spelers en het publiceren van uitslagen en standen op internet.

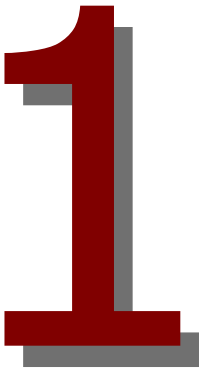
Dit project omvat het bouwen van een applicatie die de genoemde eigenschappen bevat. Het is een individueel project waarbinnen de ontwerp-, bouw- en testfasen zijn doorlopen. Daarnaast is een vooronderzoek uitgevoerd om het gehele proces betreffende het organiseren van een toernooi in kaart te brengen.

Deze applicatie ondersteund in het maken van een poule indeling, wedstrijdschema's en het doorlopen van verschillende fasen binnen een toernooi. Ook het tonen van eliminatie schema's is gerealiseerd binnen deze applicatie.

Om een goed overzicht over de applicatie te kunnen houden is deze opgedeeld in meerdere componenten. Naast de kern, die het daadwerkelijke werk uitvoert, is er een grafische user interface aanwezig die de kern aanstuurt. Daarnaast wordt er gebruik gemaakt van een component die kan communiceren met het online inschrijfsysteem voor beachvolleybaltoernooien en is er een component dat communicatie tussen meerdere instanties van de applicatie kan realiseren. Het laatste component zal echter niet binnen dit project gerealiseerd worden omdat het ontwikkelen van de andere componenten meer tijd heeft gekost dan gepland.

Op het moment van schrijven, het project loopt nog een maand door, is de meest belangrijke functionaliteit geïmplementeerd. De belangrijkste zaken die nog uitgevoerd moeten worden zijn het realiseren van het component dat communiceert met het inschrijfsysteem en het testen van het gehele systeem.

Inhoudsopgave



Scriptie

Deel 1: Achtergrond

1.1 Inleiding

Deze opdracht betreft het automatiseren van toernooien en competities voor Beachvolleybal. Deze opdracht is aangeboden door de Nederlandse Volleybal Bond (Nevobo).

De Nevobo is in 1947 opgericht en is de autoriteit en bindende factor op het gebied van volleybal en al haar facetten. In het begin draaide de bond geheel op vrijwilligers. Maar al snel werden de werkzaamheden steeds tijdrovender en complexer, waardoor in 1963 de eerste betaalde kracht aangetrokken werd. Sindsdien is de bond steeds professioneler geworden en heeft de Nevobo inmiddels 85 betaalde medewerkers.

Het belangrijkste takenpakket van de Nevobo bestaat uit:

- Het organiseren van diverse competities;
- het ondersteunen van verenigingen;
- het managen van de nationale teams;
- het stimuleren van de volleybalsport;
- het organiseren van volleybalevenementen;
- het ontwikkelen van de volleybalsport.

Om alle taken succesvol uit te kunnen voeren is de Nevobo opgedeeld in een hoofdkantoor, gevestigd in Nieuwegein, en zeven regiokantoren. Samen bedienen deze kantoren ongeveer 125.000 leden. De regiokantoren zijn vooral servicepunten in het land waar iedere vereniging en volleyballer kan aankloppen voor informatie, hulp of ondersteuning. Daarnaast hebben deze kantoren de taak om de regionale competitie in goede banen te leiden.
Het hoofdkantoor houdt zich vooral bezig met de nationale competitie, marketing, evenementen en de ontwikkeling van de sport. Tevens wordt hier de gehele financiële administratie en personeelszaken hier geregeld.

1.1.1 Nevobo en ICT

De Nevobo heeft nagenoeg geen ICTers in dienst. De ICTers die er rondlopen hebben een andere functie zoals competitiezaken. Af en toe is er wel een stagiair of afstudeerder, die een ICT project uitvoert. Maar de grote projecten laat de Nevobo vooral aan externe partijen over, iets wat voor een dergelijke organisatie helemaal geen gek idee is.

Mijn plaats, als enige persoon die zich met ICT bezighoudt, binnen de organisatie is dan ook niet duidelijk. Mijn afstudeerbegeleider is de manager van Marketing, Evenementen & Communicatie. Een afdeling waar dit afstudeerproject indirect wel iets mee te maken zal hebben maar het is geen afdeling die noodzakelijk is voor het slagen van dit project.

1.1.2 Beachvolleybal

Beachvolleybal is een variant op zaalvolleybal waar de grote verschillen vooral zitten in het feit dat er op het zand gevolleybald wordt en dat een team uit twee in plaats van zes spelers bestaat. Bij Beachvolleybal worden er doorgaans maximaal drie sets (3de set bij een gelijke stand) gespeeld.

Internationaal is Beachvolleybal een relatief jonge sport. Pas vanaf eind jaren tachtig is Beachvolleybal uitgegroeid tot een internationale topsport en mocht zich pas bij de Olympische Spelen van Atlanta in 1996 een Olympische sport noemen.
Inmiddels worden er door meer dan 200 nationale volleybalbonden activiteiten georganiseerd. Wereldwijd zijn er meer dan 2000 professionele Beachvolleyballers die onder vijf overkoepelende instanties vallen. En dat terwijl de populariteit van deze sport rustig doorgroeit.

De exacte cijfers van het aantal beoefenaars in Nederland zijn onbekend aangezien er vele toernooien worden georganiseerd die niet bij de Nevobo zijn aangesloten. Het is in ieder geval zeker dat er vele tienduizenden beachfanaten het strand elk jaar weer weten te vinden om aan één of meerdere toernooien deel te nemen.

1.2 Probleemstelling

Beachvolleybal wordt steeds populairder in Nederland. Er worden steeds meer verenigingen opgericht en ook het aantal toernooien en competities blijft groeien. Ondanks het grote aantal toernooien en competities, bestaat er nog geen goed systeem dat ondersteuning biedt bij het organiseren hiervan.

De volgende problemen doen zich voor:

- Alle toernooien worden met de hand en/of Excel documenten georganiseerd;
- Er is geen duidelijke en complete kalender met beachvolleybaltoernooien;
- Bestaande systemen van de Nevobo dienen handmatig bijgehouden te worden.

De problemen doen zich met name bij grote toernooien voor. Het toernooi in Ameland heeft bijvoorbeeld meer dan 500 teams die verspreid over 100 velden spelen. Voor een dergelijk toernooi heeft men ontzettend veel mensen nodig om alle uitslagen en standen met de hand te verwerken. Daar komt bij dat het voor zoveel teams ook nog eens veel tijd kost om wedstrijdschema's en poule-indelingen te creëren. En zo komen er nog een aantal zaken bij die steeds lastiger worden naarmate het aantal teams in een toernooi groeit.

Voor de kleinere, minder bekende, toernooien is het lastig om goed op de kaart te komen naast de bekende toernooien binnen de regio. Er zijn veel mensen die wel zin hebben om in hun eigen buurt mee te doen aan een toernooitje, maar geen zin hebben om het halve land door te reizen voor een paar potjes volleybal. Toch is het belangrijk dat deze mensen wel weten waar en wanneer de toernooien binnen hun regio zijn.
Ook de Nevobo heeft baat bij een goede toernooikalender om er op die manier achter te komen waar er al veel gevolleybald wordt en waar er als bond nog veel winst te halen valt.

Tenslotte maakt de Nevobo gebruik van een inschrijf- en rankingsysteem[1] voor de beachvolleybalcompetitie die de ere-, 1ste- en 2de divisie omvat. Dit systeem dient echter handmatig bijgehouden te worden aangezien er geen toernooisoftware is die hiermee kan communiceren. Nu valt dit voor de divisies wel mee aangezien het daar om enkele tientallen teams gaat. Bij het NK jeugd gaat het echter al snel om 125 teams die meerdere malen ingevoerd dienen te worden.
Voor de lagere niveaus is het ondenkbaar om met dit systeem te werken aangezien het dan al snel over honderden teams gaat die men handmatig moet gaan verwerken.

[1] Zie bijlage 1.

1.3 Onderzoeksvragen

Om een goed beeld te krijgen van het daadwerkelijke probleem en het vinden van de oplossing daarvan zijn er enkele onderzoeksvragen te stellen. Deze zullen tevens hulp bieden bij het formuleren van de opdracht. Tijdens de uitvoering van het project bieden deze vragen ondersteuning en is het gedurende het project altijd mogelijk om te toetsen of men wel met het daadwerkelijke probleem bezig bent.

1.3.1 Hoofdvraag

Als we de probleemstelling bekijken komen we op de volgende punten terecht:

- Handmatig een toernooi/competitie organiseren kost veel tijd;
- online zijn er geen uitslagen beschikbaar;
- er ontbreekt een volledige toernooikalender;

Met deze punten is de volgende hoofdvraag te formuleren:
"Hoe kan de Nevobo het organiseren van toernooien en competities automatiseren zodat er veel tijd bespaard blijft met het maken van wedstrijdschema's en er een koppeling ontstaat met de website beachcompetitie.nl voor het automatisch uitwisselen van centraal opgeslagen gegevens."

1.3.2 Deelvragen

- Hoe wordt een toernooi/competitie georganiseerd?
- Waar zitten de ergernissen bij het organiseren van een toernooi?
- Wat kan de website beachcompetitie.nl voor een geautomatiseerd systeem betekenen?
- Wat voor koppeling dient er met de website beachcompetitie.nl te zijn?
- Aan welke technische (infrastructuur) eisen moet de automatisering voldoen?
- Wat zijn de beste technieken voor het realiseren van een geautomatiseerd systeem?

1.4 Opdracht

Deze afstudeeropdracht bestaat uit het maken van een geautomatiseerd systeem dat de mogelijkheid bied tot het organiseren van competities en toernooien voor beachvolleybal. Dit systeem bevat elementen die specifiek bij beachvolleybal horen en is geschikt voor alle beachvolleybalniveaus.

1.4.1 Toernooien

Het systeem moet op basis van het aantal beschikbare velden, het aantal niveaus en deelnemers (per niveau) wedstrijdschema's kunnen genereren. Hierbij moet er rekening gehouden worden met een maximum wachttijd tussen wedstrijden en moet elk niveau zo veel mogelijk op een vaste groep velden spelen zodat er tussen wedstrijden door geen nethoogtes veranderd hoeven te worden. Aan elke wedstrijd zal ook een scheidsrechter toegewezen moeten worden.

Na elke wedstrijd moeten uitslagen eenvoudig in te voeren zijn zodat de standen per poule of volgende wedstrijden (afhankelijk van het wedstrijdsysteem) te genereren zijn.

1.4.2 Competities

Naast het kunnen organiseren van toernooien moeten verenigingen eenvoudig competities kunnen opzetten. Daar moet rekening gehouden worden met vaste competitiedagen, op welke velden er competitie gespeeld zal worden en met verschillende niveaus binnen de vereniging. Binnen de competitie zullen er ook rankings bijgehouden moeten worden.

1.4.3 Internet en lokaal

Er dient een koppeling te zijn met de website http://beachcompetitie.nl. Deze website bevat een inschrijfsysteem voor toernooien en het systeem zal deze gegevens moeten overnemen. Ten tijde van een internetverbinding dient het systeem ook alle gegevens gedurende het toernooi weer op te slaan in de database van de website.
Omdat de meeste beachvolleyballocaties niet over internet beschikken moet het systeem ook lokaal kunnen werken. In dat geval zal het systeem alle benodigde gegevens moeten downloaden en na het toernooi (of de competitie) weer terugsturen.

De koppeling aan de kant van de website http://beachcompetitie.nl zal gebouwd worden door een extern persoon die ook de complete website http://beachcompetitie.nl heeft gebouwd en beheerd.

1.4.4 Gebruikersonderzoek

Er zal enig onderzoek uitgevoerd moeten worden om het gewenste resultaat te behalen. Er zal enige communicatie moeten zijn met verenigingen en competitieleiders om de ergernissen van het organiseren van een toernooi aan te pakken. Daarnaast dient dit een goede lijst met business rules op te leveren zodat er met alle mogelijke situaties rekening gehouden kan worden.

1.5 Uitgangssituatie

De uitgangssituatie is een eenvoudig te besturen applicatie voor het organiseren van toernooien en competities. Deze applicatie dient te werken op zowel Windows als Mac OS en, indien mogelijk, op zoveel mogelijk andere operating systems.

Om het voor iedereen zo eenvoudig mogelijk te houden wordt er gebruik gemaakt van het huidige inschrijfsysteem van de Nevobo. Hier kan een toernooi organisator een toernooi met betreffende gegevens aanmaken waarna de deelnemers zich kunnen inschrijven. Hierna wordt de applicatie opgestart en worden alle gegevens van de website opgehaald. Vanaf dat moment zou het hele toernooi lokaal te draaien moeten zijn en kunnen de uitslagen en standen na het toernooi, of als er internet aanwezig is tijdens het toernooi, weer naar de website gestuurd worden. Omdat het netwerk uit kan vallen dienen de computers zich ten tijde van netwerkverbinding bij elke wijziging te synchroniseren zodat er overal met zo veel mogelijk gegevens gewerkt kan worden. Globaal ziet het er dan zo uit:

1. Inschrijfsysteem: Toernooi aanmaken
2. Inschrijfsysteem: Deelnemers inschrijven
3. Toernooiapplicatie: Gegevens ophalen
    a. Indien er op locatie geen internet beschikbaar is kan deze stap thuis uitgevoerd worden en worden de gegevens lokaal opgeslagen.
4. Toernooiapplicatie: Toernooi organiseren
    a. Poules (laten) indelen
    b. Wedstrijdschema's (laten) indelen en uitprinten
    c. Uitslagen invoeren
    d. Standen berekenen

    * Indien men bij een toernooi met meerdere computers werkt wordt er na alle bovengenoemde stappen gesynchroniseerd.
    * Indien er internet aanwezig is worden de gegevens ook meteen op de website gepubliceerd.
5. Toernooiapplicatie: Gegevens naar website sturen
    a. Indien er op locatie geen internet beschikbaar is kan deze stap wederom thuis uitgevoerd worden.

Naast eenvoud zijn ook flexibiliteit en functionaliteit belangrijke punten in de applicatie. De applicatie moet helpen bij de toernooien met honderden teams en dan moet er met alle mogelijkheden rekening worden gehouden. Ook als de internet- of netwerkverbindingen uitvallen dient de applicatie gewoon door te werken.

2

Scriptie

Deel 2: Analyse van de opdracht

2.1 Aanpak

Om de opdracht met succes te laten slagen zijn er verschillende fasen doorlopen:

1. Vooronderzoek;
2. Technisch- en Functioneel Ontwerp;
3. implementatie;
4. testen.

2.1.1 Vooronderzoek

Het doel van het vooronderzoek is om erachter te komen hoe een beachvolleybaltoernooi wordt georganiseerd en welke technische eisen daarbij komen kijken. Het volledig begrijpen van de organisatie achter een toernooi is essentieel voor het maken van een optimaal Functioneel- en Technisch ontwerp.

2.1.2 Technisch- en Functioneel Ontwerp

In principe heeft de Nevobo de vrije hand gegeven voor het maken van een krachtige applicatie die toernooien kan organiseren. Omdat het een individueel project is, is het niet efficiënt om een uitgebreid technisch ontwerp te maken. Een klassendiagram zou voor het overzicht in principe voldoende moeten zijn.

Uit tijdsoverweging is het Functioneel Ontwerp zo klein mogelijk gehouden en is de meeste aandacht vooral naar schermontwerpen uitgegaan. Uit de schermontwerpen moeten de use-cases duidelijk genoeg zijn om een beeld te kunnen scheppen bij alle mogelijkheden die de applicatie moet gaan bieden.

2.1.3 Implementatie

De tijdsbesparingen bij het Functioneel- en Technisch Ontwerp leverden, zoals verwacht, extra tijd op bij het implementeren. Tijdens het bouwen blijkt uiteindelijk dat niet alle ideeën het gewenste effect hebben. Bijvoorbeeld dat er bepaalde dingen niet mogelijk zijn of dat er bepaalde dingen onhandig werken. Het kostte echter heel wat minder tijd om dergelijke probleempjes tijdens het bouwen op te lossen of uit te werken dan om voor elke use case verschillende diagrammen uit te werken.

2.1.4 Testen

De testfase bestaat vooral uit het draaien van toernooien op de applicatie. Aangezien het beachvolleybalseizoen pas halverwege mei begint is het lastig om volledige toernooien te testen. Tijdens het bouwtraject moest er dan ook met verzonnen data gewerkt worden. De echte tests konden pas halverwege mei beginnen.

2.2 Vooronderzoek

Het vooronderzoek bestaat voornamelijk uit het uitzoeken hoe een beachvolleybaltoernooi van begin tot eind wordt georganiseerd. Daarnaast is het belangrijk om te weten aan wat voor eisen een applicatie zou moeten voldoen. Op het strand zijn namelijk minder faciliteiten qua stroomvoorziening etc. beschikbaar dan in een sporthal.

Het vooronderzoek is uitgevoerd door gesprekken te voeren met competitie- en wedstrijdleiders van zowel de nationale competitie als recreatieve toernooien. Deze gesprekken hebben vooral de benodigde informatie opgeleverd over alles wat er achter de schermen gebeurd bij een toernooi.

2.2.1 Toernooi organisatie

De manier waarop een toernooi wordt georganiseerd is sterk afhankelijk van het aantal teams dat meedoet. Er zullen ook andere factoren meespelen, zoals de faciliteiten en het aantal beschikbare velden. Echter hebben deze factoren een directe link met het aantal teams dat zal deelnemen. Hoe meer deelnemers, hoe beter de faciliteiten en hoe meer velden er aanwezig zullen zijn. Beter kan gezegd worden dat het maximum aantal deelnemers afhankelijk is van de faciliteiten van de accommodatie.

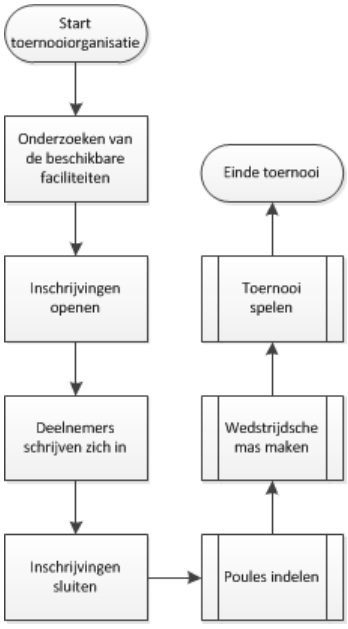
2.2.1.1 De basis

In principe wordt elk toernooi op dezelfde manier georganiseerd:

1. Er wordt besloten dat er een toernooi georganiseerd gaat worden;
2. er wordt vastgesteld op hoeveel velden en met hoeveel niveaus er zal worden gespeeld;
3. de inschrijvingen worden geopend;
4. deelnemers schrijven zich in;
5. de inschrijvingen worden gesloten;
6. de poule indelingen worden gemaakt;
7. het wedstrijdschema wordt gemaakt;
8. het toernooi kan worden gespeeld.

Uiteraard komt er meer kijken bij het organiseren van een toernooi. Bijvoorbeeld het in orde brengen van de velden (nethoogtes, lijnen etc.), het regelen van de catering, het regelen van de infrastructuur etc. Echter zijn dergelijke zaken niet relevant voor dit project omdat deze niet geautomatiseerd kunnen worden.

Het organiseren van een toernooi is redelijk recht toe recht aan en wordt dan ook steeds op (ongeveer) dezelfde manier gedaan. Hierdoor is het ook goed mogelijk om een toernooi te automatiseren omdat de software het proces eenvoudig zou moeten kunnen volgen.



Afbeelding 1: Proces toernooi organisatie

Binnen het proces worden er wel een paar complexe stappen uitgevoerd. Voornamelijk de poule indelingen en het wedstrijdschema zijn de tijdrovende stappen. Hier dient de organisator namelijk na te gaan denken over de beste indelingen en schema's. Want waar wordt een dergelijke poule indeling op gebaseerd? En wat is het meest optimale wedstrijdschema? Deze twee stappen die verschillen tussen een recreatief toernooi en een toernooi dat meetelt voor de nationale ranking. Dit verschil wordt mede bepaald door het aantal deelnemende teams en de faciliteiten.

2.2.1.2 Klassen en Fasen

De eerste stap die gewoonlijk ondernomen wordt is het vaststellen van het systeem waarmee gespeeld zal worden. Met systeem worden geen computerapplicaties of hulpmiddelen bedoeld maar met welke klassen en fasen gespeeld zal worden.

Voordat de inschrijvingen worden geopend moet het voor potentiële deelnemers duidelijk zijn voor welke niveaus (klassen) zij zich kunnen inschrijven. Afhankelijk van het toernooi worden er voor of na de inschrijvingen de bijbehorende fasen vastgesteld. Binnen een fase staat vast met welk type wedstrijdschema er wordt gespeeld, hoeveel poules er zijn en hoeveel teams er maximaal in één poule kunnen deelnemen.
Na het spelen van een fase kan er een ranglijst per poule worden opgemaakt en is het mogelijk om bepaalde teams naar eventuele volgende fasen door te laten stromen.

Bijvoorbeeld:
Bij de klasse 2x2 Hoog wordt er eerst gespeeld volgens een poule systeem. In dit systeem draaien acht poules van ieder vier teams. Dat resulteert in een wedstrijdschema waar elk team binnen elke poule één keer tegen elkaar speelt.
Wanneer alle wedstrijden zijn gespeeld wordt er per poule een stand opgemaakt en gaan van alle poules de eerste twee teams door naar een volgende fase. De overige teams zijn klaar met het toernooi.
In de volgende fase wordt gespeeld met een eliminatie systeem. Dus wanneer een team een wedstrijd verliest, ligt dat team uit het toernooi. In dit eliminatie systeem zijn 16 teams (de nummers één en twee van de acht poules) die het tegen elkaar opnemen. Beginnend bij de 1/16de finales daarna de 8ste finales, daarna de kwartfinales, halve finales en uiteindelijk de finale.
Na deze fase kan een eindstand voor de hele klasse worden opgemaakt. De winnaar van de finale wordt eerste, de verliezer 2de. De verliezers van de halve finales delen de 3de plek en de verliezers van de kwartfinales delen de 5de plek. Op deze manier kan doorgaan worden tot aan de 1/16de finales.

Bovenstaand voorbeeld is slechts één uit een oneindig aantal combinaties van fasen. Zo wordt er ook wel eerst met een eliminatie systeem gewerkt, dan met een poule systeem en vervolgens weer met een eliminatie systeem. Een poule systeem kan ook gevolgd worden door een tweede poule systeem hebben voor de kruisfinales enz.

Bij Beachvolleybaltoernooien wordt er doorgaans met drie verschillende typen wedstrijdschema's gewerkt:

- Een poule schema, iedereen in de poule speelt tegen elkaar;
- Single elimination, de teams binnen een poule worden aan elkaar gekoppeld. De winnaar gaat door naar de volgende ronde, de verliezer is klaar met het toernooi;
- Double elimination, hetzelfde principe als single elimination alleen krijgt de verliezer een tweede kans door naar de zogenoemde losers bracket verplaatst te worden. Hiervandaan is het nog steeds mogelijk om de finale te bereiken.

2.2.1.3 Poule indeling

De poule indeling is dan nog de gemakkelijkste stap. Binnen een recreatief toernooi kan deze in principe willekeurig ingedeeld worden. Bij een klein toernooi waar men de teams kent zou men eventueel rekening met de sterkte van de teams kunnen houden. En deze dan juist bij elkaar zetten voor leuke wedstrijden of juist apart houden om de teams van vermoedelijk gelijke niveaus tegen elkaar te zien spelen in de finalerondes.

Bij de jeugd telt de leeftijd van de deelnemers mee. Dan is het belangrijk dat mensen van gelijke leeftijd tegen elkaar spelen omdat deze meestal ook hetzelfde niveau hebben. Daarnaast speelt de lengte ook een rol. Een kind van 14 is hoogstwaarschijnlijk een stuk groter dan een kind van 12 en zal daarom een stuk minder moeite hebben om een bal over het net te spelen. Bij grote toernooien zal de jeugd zich zelf voor de juiste leeftijdscategorie moeten aanmelden. Bij lokale toernooien waar gewoonlijk jeugd van de vereniging naartoe komt weet de organisator meestal wel hoe oud de deelnemer is en deelt die vervolgens in de juiste categorie in.

Hoewel de poule indeling een gemakkelijke stap is, is het wel een tijdrovende stap. Bij het maken van de indeling wordt eerst bedacht of er in 4 poules van 3 teams of juist in 3 poules van 4 teams of een andere combinatie van aantal poules en teams per poule gespeeld gaat worden. Daarna worden teams toegewezen aan poules.
Bij een klein toernooi (25 – 50 teams) is dit in een uurtje gedaan. Maar bij grote toernooien kan dit wel een halve dag in beslag gaan nemen.

Nationale competitie

In de nationale beachcompetitie zijn de poule indelingen eigenlijk nog eenvoudiger uit te voeren. De nationale competitie hanteert namelijk regels voor de poule indeling die eenvoudig te volgen zijn. De poule indelingen van de competitie zijn namelijk gebaseerd op de zogenoemde seeding van de deelnemers. De seeding vertelt welke deelnemer waarschijnlijk eerste zal worden, welke deelnemer waarschijnlijk tweede zal worden enzovoorts. Deze seeding wordt bepaald door te kijken op welke plaatsen de deelnemer tijdens zijn laatste vier toernooien is geëindigd.

Stel, deelnemer A wordt een keer 1ste, een keer 2de, nog een keer 1ste en een keer 4de. Dan worden deze bij elkaar opgeteld en door vier gedeeld: 1 + 2 + 1 + 4 = 8 / 4 = 2. Deelnemer A heeft dan dus een seeding van 2. Als deelnemer B 2de, 1ste, 2de en 3de wordt, dan heeft hij een seeding van 2 + 1 + 2 + 3 = 7 / 4 = 1,75. Oftewel men verwacht dat deelnemer B hoger zal eindigen dan deelnemer A.
Op deze manier is het mogelijk om een complete lijst te maken van deelnemers en de uiteindelijke plek die ze naar verwachting halen. Bijvoorbeeld de lijst in Afbeelding 2.

Deelnemer	Seeding
Jan	1
Piet	2
Kees	3
Hans	4
Tim	5
Chris	6
Wouter	7
Patrick	8
Bas	9

Afbeelding 2: Lijst van deelnemers met seeding

Deze negen deelnemers zullen vervolgens op de volgende manier over een bepaald aantal poules worden verdeeld: Te beginnen bij Poule A waar degene met de hoogste seeding in wordt geplaatst. In het geval van Afbeelding 2 is dat Jan. Daarna wordt in Poule B degene met de eerstvolgende seeding in wordt geplaatst (Piet). Zo gaat het verder tot in iedere poule één deelnemer geplaatst is. Vervolgens word de volgende deelnemer wederom in de laatste pool geplaatst en alle andere deelnemers aflopend naar de eerste poule. Zolang er nog deelnemers over zijn worden deze vervolgens weer oplopend tot de laatste poule ingedeeld en weer terug.

Poule A	Poule B	Poule C
Jan	Piet	Kees
Chris	Tim	Hans
Wouter	Patrick	Bas

Afbeelding 3: Voorbeeld van een poule indeling met 3 poules

Kortom, de poules worden zigzaggend ingedeeld. Een voorbeeld met 3 poules is te zien in Afbeelding 3. Hier is Jan (seeding 1) ingedeeld in Poule A, Piet (seeding 2) in Poule B, Kees (seeding 3) in Poule C, Hans (seeding 4) in Poule C, Tim (seeding 5) Poule B, Chris (seeding 6) Poule A etc.

De theorie achter een dergelijke poule indeling is dat de sterkste teams de meeste kans hebben om elkaar pas in de finale te treffen en er niet al in de poulefase uit zullen vliegen. Deze theorie is overigens niet uniek. Het WK Voetbal 2010 werkt op een soortgelijke manier waar de hoogste zeven teams in de Fifa ranking verspreid worden over alle poules zodat deze elkaar pas in de finalerondes kunnen treffen.

2.2.1.4 Wedstrijdschema's

Het maken van wedstrijdschema's is een wat lastiger verhaal. Hier zijn voor zowel recreatieve toernooien en de competitie niet echt regels of richtlijnen voor. De bedoeling is uiteraard om de schema's zo optimaal mogelijk te maken.

Om alle wedstrijden binnen de begin- en eindtijden van het evenement te houden dienen de wedstrijden een maximale lengte te hebben. Bij recreatieve toernooien is dat meestal geen probleem aangezien het daar gebruikelijk is om op tijd te spelen, de lengte van een wedstrijd is eenvoudig aan te passen aan de hand van het aantal wedstrijden. Maar bij de competitie dienen er altijd twee of drie sets van 21 punten (3de set 15 punten) met twee punten verschil gespeeld te worden. In theorie zou een wedstrijd dus in tien minuten klaar kunnen zijn, maar het kan ook twee uur duren. Gelukkig wijst de praktijk uit dat een volledige wedstrijd meestal

niet langer dan 50 minuten duurt. Een lengte die dan ook algemeen aangenomen wordt. Voor dergelijke toernooien is dan ook een harde limiet van aantal teams dat kan deelnemen.

Als de poule indelingen bekend zijn en daarmee het aantal te spelen wedstrijden bekend is, is het nog niet klaar. Als de wedstrijden domweg één voor één worden ingedeeld en Piet daarmee om negen uur mag beginnen, zijn tweede wedstrijd pas om één uur heeft en zijn laatste wedstrijd om vier uur, dan is Piet niet blij en is het schema ook niet optimaal te noemen. Het doel is dus om iedereen zo veel mogelijk wedstrijden te laten spelen in zo'n kort mogelijke periode. Toch is het moeilijk om pauzes tussen wedstrijden te voorkomen en dat is ook niet gewenst. Iedereen wil na twee wedstrijden namelijk wel wat water drinken of een broodje eten. Maar het blijft gewenst om de pauzes zo kort mogelijk te houden.

Afhankelijk van het aantal beschikbare velden en het aantal wedstrijden is het al mogelijk om een redelijk goed wedstrijdschema te genereren als er met het bovenstaande rekening gehouden wordt. Als er relatief weinig beschikbare velden zijn zal het al snel voorkomen dat bepaalde niveaus vroeg op de dag spelen en dat andere niveaus laat op de dag spelen. Niveau A speelt dan bijvoorbeeld tussen negen en twaalf uur met daarna de finales terwijl niveau B van twaalf tot vier uur speelt.
Als er relatief veel velden beschikbaar zijn dan is er veel meer mogelijk en kan er beslist worden dat een niveau ook de hele dag zal duren. Meestal spelen de teams ook meer wedstrijden wanneer er veel velden beschikbaar zijn.

In de competitie kan er met wat extra zaken rekening gehouden worden. Zo gaat men er bijvoorbeeld van uit dat de teams met seeding één en twee in de finale terecht zullen komen. Daar de finale gewoonlijk de laatste wedstrijd is, is het een optie om teams wat later te laten beginnen.

Jeugdspelers spelen het liefst zoveel mogelijk wedstrijden en zitten niet zo op pauzes (op een enkele pauze na) te wachten. In sommige recreatieve toernooien is het ook toegestaan dat jeugdspelers zich eerst bij de jeugd inschrijven en ook nog eens bij de senioren. Dat gebeurt meestal wanneer beide niveaus op andere tijden spelen, bijvoorbeeld de jeugd 's ochtends en de senioren 's middags.
In de competitie is het voor jeugdspelers toegestaan om mee te doen met hogere niveaus. Zo kan het op het NK Jeugd voorkomen dat een team van twee spelers van 13 jaar meedoet met de categorie jonger dan 14 en met de categorie jonger dan 16 en ook nog eens met de categorie jonger dan 18. Alle niveaus op het NK Jeugd spelen de gehele dag en daarbij moet voorkomen worden dat een team twee wedstrijden op hetzelfde tijdstip heeft.

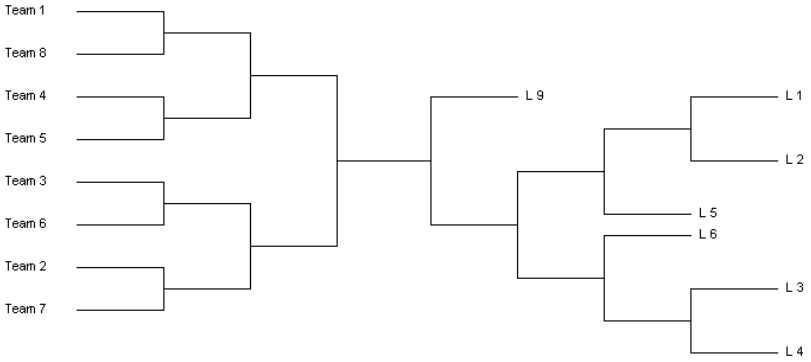
Het maken van wedstrijdschema's kan zo complex gemaakt worden als men wil. Op een hele simpele manier is het al mogelijk om een kloppend schema te maken. Maar naarmate men met meer zaken rekening wil gaan houden hoe ingewikkelder het zal worden. Bij het automatiseren van de wedstrijdschema's dient er rekening gehouden te worden met deze complexiteit zodat de applicatie ook elk evenement gebruikt kan worden. Dat maakt het ontwikkelen van een dergelijke applicatie extra ingewikkeld.

Wedstrijdschema's & Single- en Double elimination

Bij het maken van de wedstrijdschema's komt nog iets extra's kijken in het geval er met single- of double elimination wordt gespeeld. Deze typen schema's worden vooral gebruikt in de nationale competitie.

Omdat niet alle teams tegen elkaar zullen spelen bij deze systemen is het belangrijk om de juiste teams met elkaar te koppelen. Dit wordt gedaan aan de hand van de seeding. Om als doel te houden dat de sterkste teams in de finale terecht zullen komen moeten deze aan de uiteinden van het schema ingedeeld worden.
Als regel geldt dat de seeding van de teams die in de eerste ronde tegen elkaar spelen opgeteld het totale aantal teams + 1 is. Bij acht teams is de optelsom van de seeding van elk koppel dus negen. Bij 16 teams is deze som dus 17.

Team één speelt dus tegen team acht, team twee tegen team zeven, team drie tegen team zes en team vier tegen team vijf. Door deze vervolgens om de juiste positie te zetten kunnen de nummers één en twee elkaar pas in de finale ontmoeten.



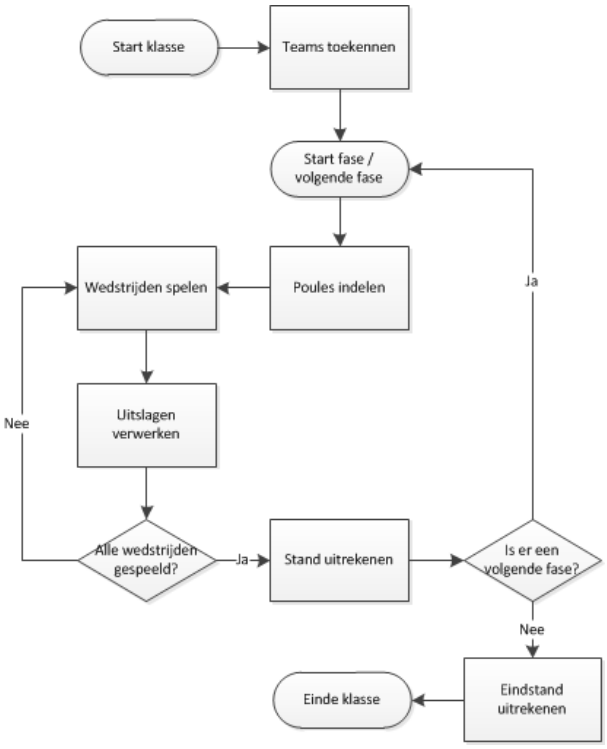
Afbeelding 4: Double elimination met acht teams

2.2.1.5 Het spelen van een toernooi

Wat in het bovenstaande proces niet word vermeld is dat er tijdens het spelen van het toernooi ook nog een paar zaken dienen te gebeuren. De deelnemers willen uiteraard graag zien op welke plek ze geëindigd zijn en dat is alleen mogelijk door de uitslagen van de wedstrijden te verwerken. Ook komt het geregeld voor dat er na de wedstrijden binnen een poule nog finales gespeeld moeten worden. Iets wat zonder de uitslagen van de wedstrijden ook niet mogelijk is. Deze zijn immers nodig om te weten welke teams in welke finale ingedeeld moeten worden.

Zoals gezegd kan een klasse (niveau) uit verschillende fasen bestaan. Deze fasen werken ieder met hun eigen type wedstrijdschema en bestaan weer uit een aantal poules. Omdat er in principe een oneindig aantal klassen, fasen en poules kunnen bestaan kan er gesproken worden over een iteratief proces zoals te zien in Afbeelding 5.

Bij het begin van de klasse worden er teams aan toegekend. Dat is iets wat tijdens de voorbereiding (na de sluiting van de inschrijvingen) al kan gebeuren. Daarna start de eerste fase waarbinnen de poules ingedeeld zullen worden. In principe is tijdens de voorbereiding al vastgelegd welke teams in welke poule terecht zullen komen. Dit is dan ook slechts een administratieve taak.



Afbeelding 5: Proces spelen toernooi

Na de poule indeling kunnen de wedstrijden van het wedstrijdschema worden gespeeld. Na elke wedstrijd zal de uitslag doorgegeven worden aan de wedstrijdleiding. Deze verwerkt de uitslagen en kijkt of alle wedstrijden van de fase zijn gespeeld. Indien dit niet het geval is gaan we terug naar het spelen van de wedstrijden.
Zijn alle wedstrijden gespeeld? Dan kan de stand van elke poule berekend worden.
Uiteraard is het mogelijk om na het verwerken van de uitslagen ook tussenstanden uit te rekenen. Deze stap kan redelijk hectisch verlopen omdat alle uitslagen tegelijk worden ingeleverd. Als er 50 wedstrijden tegelijk worden gespeeld is dat nogal wat om te verwerken. Hier komt ook bij kijken dat de stand van een groot aantal poules in een zeer korte tijd uitgerekend moet worden.

Wanneer de stand van de fase bekend is kan er of een volgende fase worden gestart zoals bijvoorbeeld de finales. Of kan de eindstand van de klasse worden berekend.

De enige opmerking die hier bij gemaakt moet worden is dat er bij het eindigen van een fase meerdere nieuwe fasen kunnen worden gestart. Bijvoorbeeld een fase om de nummers één van elke poule tegen elkaar te laten spelen en een fase om alle nummers twee van elke poule tegen elkaar te laten spelen.

2.2.1.6 Conclusie

Er kan gezegd worden dat het organiseren van een toernooi in principe een redelijk simpel proces is. Tijdens elke stap is het duidelijk wat er moet gebeuren en hoe dat aangepakt dient te worden.

De complexiteit van een toernooi ligt hem vooral in de vele (oneindige) mogelijkheden die gecreëerd kunnen worden. Deze complexiteit is vooral terug te vinden bij het maken van de wedstrijdschema's waar uit de talloze combinaties de meest optimale gekozen dient te worden. Uiteindelijk heeft de organisatie zelf in de hand hoe complex men het toernooi wil maken. Bij het automatiseren van een toernooi dient de ontwikkelaar echter met de meest complexe situaties rekening te houden.

2.2.2 Eisen aan een eventuele applicatie

De meeste beachvolleybalaccomodaties staan meestal op locaties ver weg van stroom, water en internet. Op vaste accomodaties die het hele jaar blijven staan is meestal nog wel stroom en water te vinden maar blijft internet een probleem. Op de stranden waar immense toernooien worden georganiseerd (bijvoorbeeld Scheveningen en Ameland) is het een stuk lastiger om dit soort zaken te realiseren.

Water en computers gaan sowieso niet zo goed samen dus dat vormt geen probleem voor een applicatie. Wanneer het een groot toernooi betreft en er geen vaste stroomlijn aanwezig is wordt er gebruik gemaakt van aggregraten.
Het probleem van aggregraten is echter dat ze niet altijd even stabiel zijn. Ze kunnen stukgaan waarmee de gehele stroomvoorziening wegvalt. Voor een applicatie is dat niet meteen een ramp aangezien deze in alle gevallen op een laptop zal draaien die gewoonlijk een accu heeft om zulke gevallen op te vangen. Maar accu's die een paar jaar oud zijn gaan niet zo lang mee waardoor de laptop uitvalt voordat een gebruiker ook maar de kans heeft gekregen om de gegevens op te slaan. Er dienen dus regelmatig backups weggeschreven te worden voor het geval de stroom uitvalt.

Is het een groot toernooi? Dan zal er waarschijnlijk gebruik gemaakt worden van meerdere computers. Het Beachvolleybaltoernooi in Ameland is zo groot dat een rondje om het eiland lopen bijna sneller is om van veld 100 bij veld één te komen. Daar is het dus geen optie om een centraal punt te gebruiken om alle uitslagen te verwerken.
Meerdere computers betekent een netwerkverbinding en dus netwerkondersteuning binnen de applicatie. Dat is allemaal nog niet zo spannend. Maar wat als het aggregraat op Ameland opeens uitvalt? Hoe communiceren de computers dan met elkaar? Is er nog wel een computer die zal functioneren?

Een middel dat een nog groter probleem zal vormen is internet. Er zullen een paar uitzonderingen zijn maar over het algemeen is er geen internet aanwezig tijdens Beachvolleybaltoernooien. Toch moet er gebruik gemaakt worden van de gegevens van het inschrijfsysteem. Deze gegevens zullen dus ook offline beschikbaar moeten zijn ten tijde van het toernooi om later weer teruggestuurd te kunnen worden.

Tenslotte moet er rekening gehouden worden met verschillende besturingssystemen. De MacBook van Apple is redelijk populair en speelt zeker mee in de laptopwereld. Ook is er wedstrijdleiders gesproken die aangaven over een MacBook te beschikken. Er dient dus rekening gehouden te worden met meer dan Windows alleen.

Alle eisen op een rijtje:

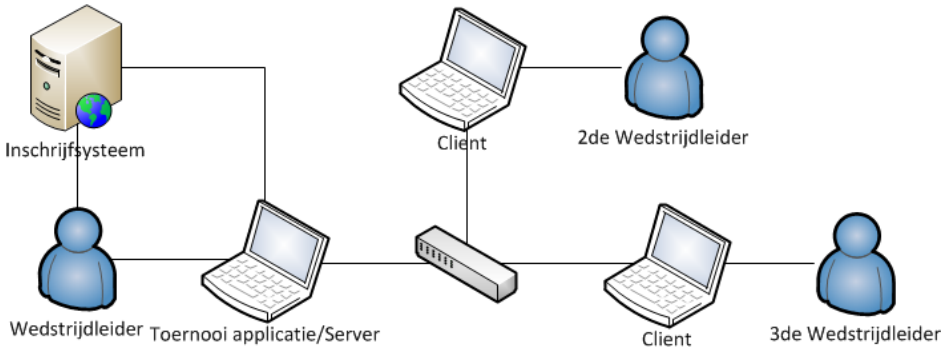
- Een eventuele applicatie moet regelmatig backups maken ivm mogelijke stroomuitval;
- Een eventuele applicatie moet ook zonder netwerk blijven functioneren en later kunnen synchroniseren met andere clients;
- Online gegevens dienen van tevoren gedownload en offline gebruikt te kunnen worden;
- Een eventuele applicatie moet op meerdere operating systems kunnen werken.

2.3 Functioneel Technisch Ontwerp

Het Functioneel Technisch Ontwerp omvat binnen dit project het klassendiagram en de schermontwerpen. Voor de communicatie met het inschrijfsysteem zal gebruik gemaakt worden van webservices. Hiervoor zijn de methoden en hun parameters vastgelegd.

2.3.1 Architectuur

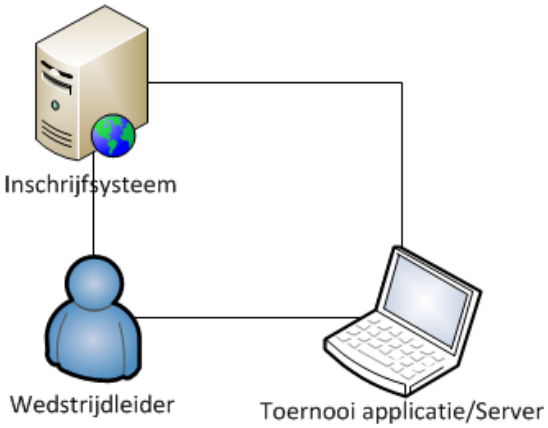
De meest ingewikkelde situatie die zich kan voordoen binnen een toernooi is te zien in Afbeelding 6. Hier is te zien dat een toernooi gebruik kan maken van meerdere computers die verbinding met elkaar hebben. Deze situatie zal zich echter niet vaak voordoen.



Afbeelding 6: Meest complexe situatie

In Afbeelding 6 *zijn slechts twee clients met bijbehorende wedstrijdleiders aanwezig. Dit kan ook één client zijn maar het kunnen ook drie zijn. Dit is afhankelijk van de situatie en er wordt in principe vanuit gegaan dat er een onbeperkt aantal clients aanwezig kan zijn.*

Een meer gebruikelijke situatie is een toernooi waar gebruik wordt gemaakt van één computer die bediend wordt door de wedstrijdleider. Eventueel heeft deze computer tijdens het toernooi verbinding met het inschrijfsysteem. Dit laatste is echter niet noodzakelijk. In Afbeelding 6 heeft men alleen te maken met het inschrijfsysteem, de wedstrijdleider en de toernooi applicatie. De switch, clients en andere wedstrijdleiders komen dan niet binnen de architectuur voor. Deze situatie is te zien in Afbeelding 7.



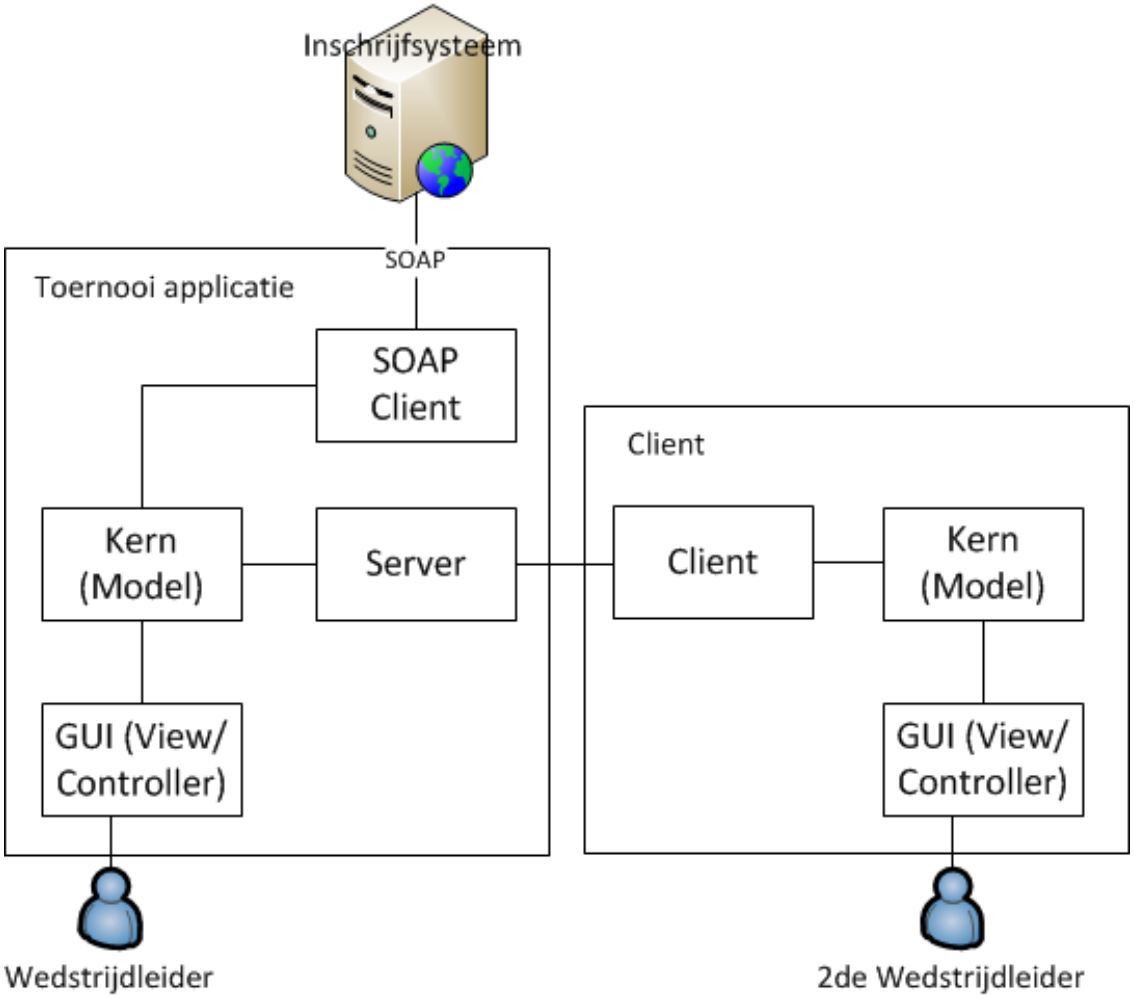
Afbeelding 7: Gebruikelijke situatie

Tenslotte zal regelmatig voorkomen dat er slechts één Wedstrijdleider en computer aanwezig zullen zijn. Wel kan het zo zijn dat de communicatie met het inschrijfsysteem dan al op een eerder moment heeft plaatsgevonden om de gegevens te downloaden.

2.3.1.1 Architectuur toernooi applicatie

De toernooi applicatie zelf bestaat uit vijf belangrijke componenten. De applicatie is met het MVC (Model-View-Controller) model in het achterhoofd opgebouwd. Alleen word er geen duidelijk onderscheid gemaakt tussen de View en Controller. De Grafische User Interface laat de gegevens van het Model zien en stuurt het Model vervolgens ook weer aan.
Omdat platform onafhankelijkheid een eis is worden alle componenten in Java geschreven.

Andere belangrijke componenten, die alleen door het Model worden aangestuurd, zijn de SOAP Client, Server en Client.



Afbeelding 8: Architectuur applicatie

SOAP Client

Het inschrijfsysteem (beachcompetitie.nl) is gebouwd in PHP en biedt nog geen mogelijkheden voor het uitwisselen van data met andere systemen. De meest eenvoudige manier om met deze website te communiceren zou via webservices zijn. Met de ontwikkelaar van dit systeem is dan ook afgesproken dat er een SOAP Webservice geïmplementeerd zou worden op de website. Ook is er meteen geïnventariseerd welke gegevens de website zou kunnen leveren en daaruit is een geschikt WSDL bestand gekomen. Er is afgesproken om de volgende methoden te hanteren:

```
int login(gebruikersnaam, wachtwoord), geeft een uniek sessie id terug.

Toernooi[] getToernooiLijst(sessieid)
Klasse[] getKlassen(Toernooi)
Team[] getTeams(Toernooi, Klasse)
Speler[] getSpelers(Team)
getWedstrijden(Klasse)
setWedstrijdUitslag(Wedstrijd, Uitslag)
setKlasseUitslag(Klasse, Uitslag)
```

Momenteel dient de toernooi applicatie met meer zaken rekening te houden dan de website, bjvoorbeeld de poule-indelingen, bepaalde regels die gehanteerd zullen worden tijdens wedstrijden etc. Dergelijke zaken zullen tijdens het importeren van gegevens ingesteld moeten worden binnen de applicatie.

Server & Client

Wanneer een toernooi met meerdere computers georganiseerd wordt dient er één als server aangewezen te worden. De overige computers functioneren dan als client. De bedoeling is dat alle computers ten tijden van een netwerkverbinding altijd met dezelfde gegevens werken. Wanneer de netwerkverbinding tijdelijk is onderbroken dienen de computers daarna te synchroniseren.

Van alle componenten is het lokale netwerk de minst belangrijke. Er zijn maar een paar toernooien waar er met meerdere computers wordt gewerkt en ook daar is het in principe mogelijk om één computer te gebruiken hoewel dat uiteraard tot extra werk leidt.
Omdat dit het minst belangrijke component is, is besloten om hier als laatste naar te kijken. Op het moment van is het nog niet duidelijk welke techniek hier voor gebruikt zal worden.

Er zijn een paar tests gedaan met twee verschillende technieken. Namelijk RMI en het gebruik van Sockets. Hoewel RMI de beste keus lijkt te zijn is het tot nu toe met Sockets eenvoudiger geweest om data op andere computers te manipuleren en om eventuele firewall problemen te omzeilen. RMI maakt namelijk gebruik van een aantal zaken in Java waar de ontwikkelaar minder invloed op heeft. Nader onderzoek moet echter nog uitwijzen of het gebruik van Sockets inderdaad de beste methode is.

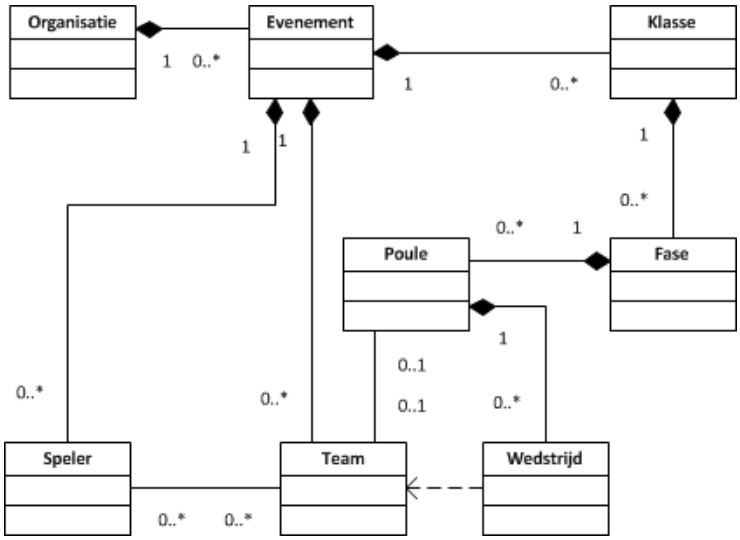
Een andere logische oplossing zou het gebruik van een application server zijn. Hierbij is er slechts één computer waar de daadwerkelijke applicatie op draait. De applicatie is dan toegankelijk via een Web GUI wat heel veel problemen zou oplossen. Er zullen namelijk minder problemen met firewalls optreden omdat alles over HTTP gaat. Daarnaast hoeft er niet gesynchroniseerd te worden tussen meerdere computers wat nog wel eens problemen zou kunnen opleveren.

Toch is het gebruik van een application server geen oplossing. Het zou betekenen dat er altijd een netwerkverbinding beschikbaar moet zijn en dat de laptop/computer waar de server op draait altijd stroom nodig heeft. Als de server niet bereikbaar is zal de applicatie namelijk op geen enkele computer meer werken. Stroomuitval (en daarbij netwerkuitval) komt op grote evenementen nog steeds voor wat zou betekenen dat het belangrijkste element van een toernooi, het volleyballen zelf, niet meer uitgevoerd kan worden. Het systeem zal dus decentraal moeten werken zodat alle laptops over de data beschikken en dat deze slechts uitslagen door hoeven te geven wanneer er netwerkverbinding beschikbaar is.

2.3.2 Klassendiagram

Het belangrijkste deel van het klassendiagram is te zien in Afbeelding 9. Het volledige klassendiagram is te vinden in Bijlage 2.

Duidelijk is dat het allemaal bij de Organisatie begint. Het lijkt mij dat er in bijna alle gevallen slechts één Organisatie aanwezig zal zijn. Echter kan het in een enkel geval voorkomen dat iemand voor bijvoorbeeld twee verenigingen evenementen organiseert.



Afbeelding 9: Klassendiagram

Het Evenement heeft met Klassen, Spelers en Teams te maken. De associatie met Spelers en Teams is slechts voor het bewaren van deze objecten wanneer ze nog aan een Klasse toegewezen moeten worden of wanneer de Klasse waar ze in zitten verwijderd wordt.

Het echte werk zal in het Klasse, Fase, Poule en Wedstrijd gedeelte moeten gebeuren. Het is de bedoeling dat met name De Fase en Poule klassen verantwoordelijk worden voor het afhandelen van het automatisch indelen van spelers en wedstrijden. Ook het afhandelen van de stand en het beëindigen van een fase wanneer alle wedstrijden zijn gespeeld moet door deze klassen worden uitgevoerd.
In het diagram is te zien dat de Wedstrijd geen directe link heeft naar de Teams die tegen elkaar spelen. In eerste instantie was dit wel een harde link, oftewel een Wedstrijd object had directe referenties naar de beide Team objecten. Bij het bouwen bracht dat echter een aantal complicaties met zich mee.

2.3.3 Schermontwerpen

Alle schermontwerpen zijn gemaakt met pen en papier. Dat was de snelste en gemakkelijkste manier. Uiteindelijk blijven het dan ruwe schetsen maar dat was voldoende om de applicatie mee te bouwen.

Mijn bedoeling was om vervolgens de echte schermen (gebouwd in Java) te bespreken met de opdrachtgever. Het is niet de meest efficiënte manier, voor deze aanpak is gekozen om de volgende twee redenen:
Ten eerste geeft het de opdrachtgever een beter beeld over het definitieve scherm en kon hij ook alvast aanvoelen hoe de applicatie globaal bestuurd zou moeten worden.
Daarnaast, wat de belangrijkste reden was, kon er meer ervaring opgedaan worden met SWT. SWT schijnt een opkomende toolkit te zijn voor Java en wordt steeds vaker gebruikt in grotere applicaties. Door eerst een aantal schermontwerpen met SWT te bouwen kon er meer kennis over SWT opgebouwd worden. Zodoende kon ook beter geëvalueerd worden of SWT geschikt is voor de te implementeren applicatie.

2.3.3.1 Speerpunten schermontwerpen

Iedereen heeft als doel om duidelijke schermen te bouwen die elke leek begrijpt en kan bedienen. Echter is usability een vak apart waar zoveel over is geschreven dat het onmogelijk is om, binnen de gestelde tijd, met alle richtlijnen en mogelijkheden rekening te houden. Door opgedane kennis als Inspecteur Webtoegankelijkheid drempelvrij.nl bij de Stichting Accessibility was het bouwen van duidelijke schermen niet erg ingewikkeld. Daarnaast is gebruik gemaakt van het boek ‘Don't Make Me Think!’ van Steve Krug voor extra achtergrond informatie.

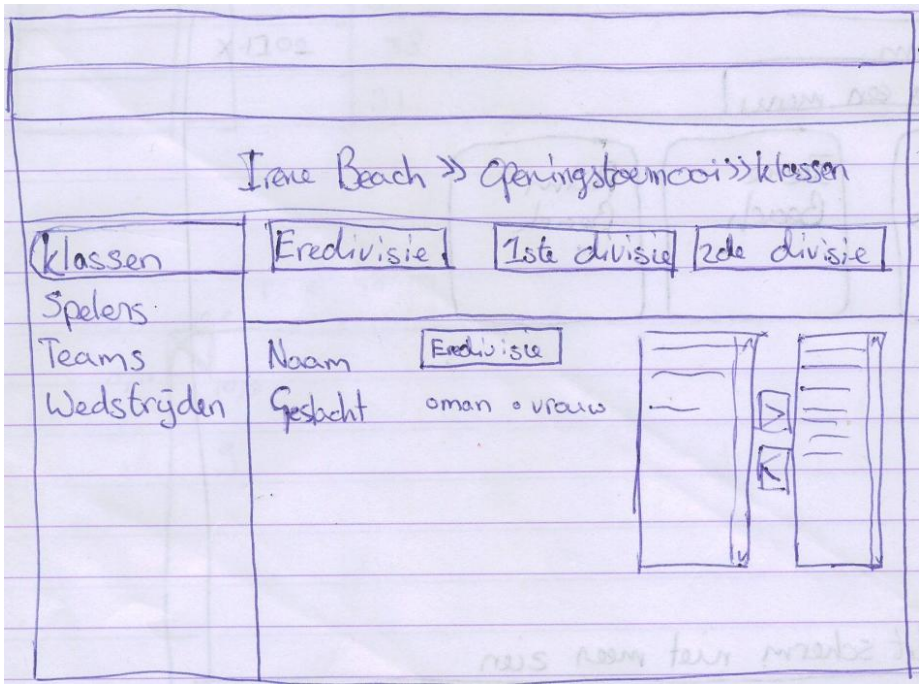
Het belangrijkste is dat het scherm zelf duidelijk moet maken wat de bedoeling is en dat er geen lappen tekst nodig zijn om dit uit te leggen. Gewoonlijk scannen mensen een scherm zonder het daadwerkelijk te lezen. Alleen essentiële tekst, zoals een label bij een tekstveld, zou meteen getoond moeten worden. Uitgebreide informatie zou getoond kunnen worden door middel van tooltips.

Omdat de applicatie voornamelijk op het strand gebruikt zal worden zal er rekening gehouden moeten worden met het contrast. Door zonlicht ziet men minder op een computerscherm. Daarom is de leesbaarheid belangrijker dan het uiterlijk.

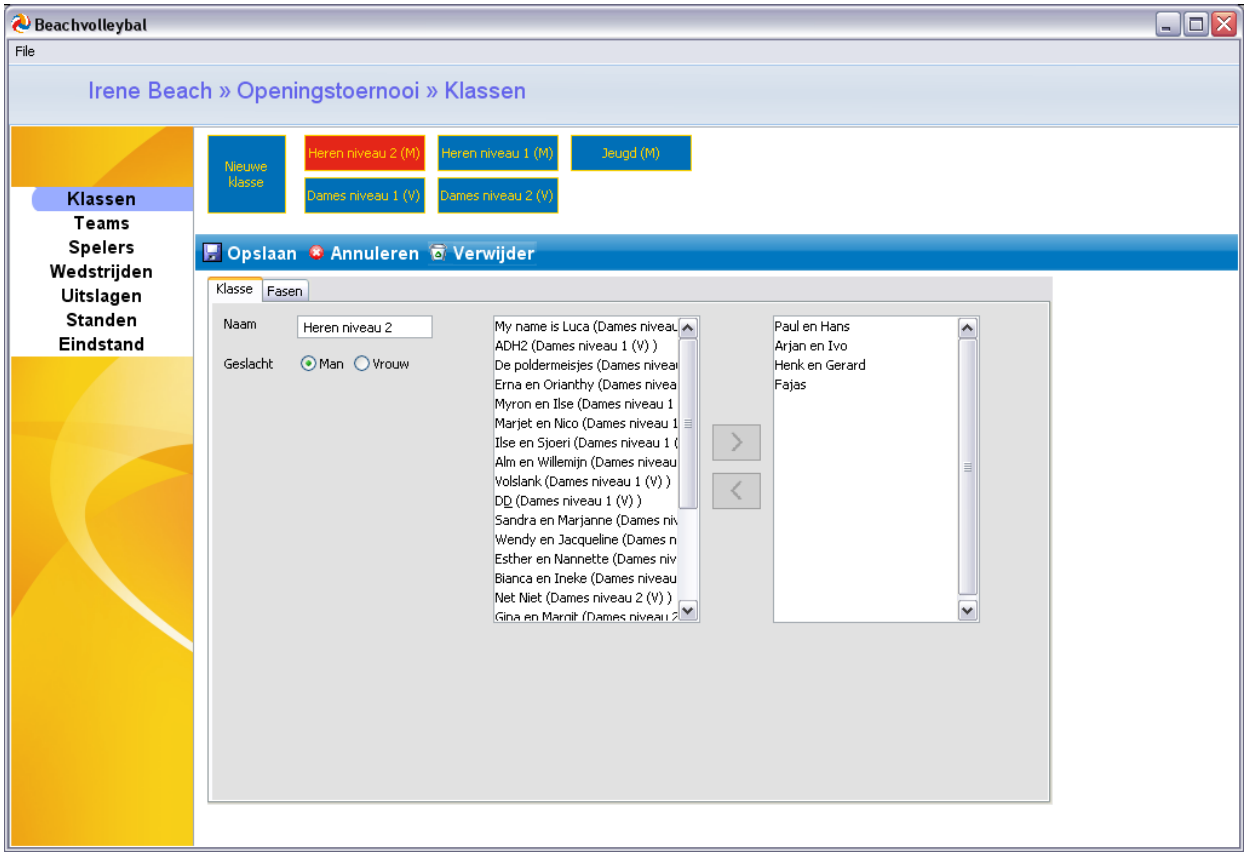
Tenslotte moeten knoppen en labels duidelijk aangeven waarvoor ze zijn bedoeld en wat er gebeurt als er op gedrukt wordt. Dit om de duidelijkheid bij het navigeren door de applicatie te bevorderen.

2.3.3.2 Schermontwerpen en realisatie daarvan

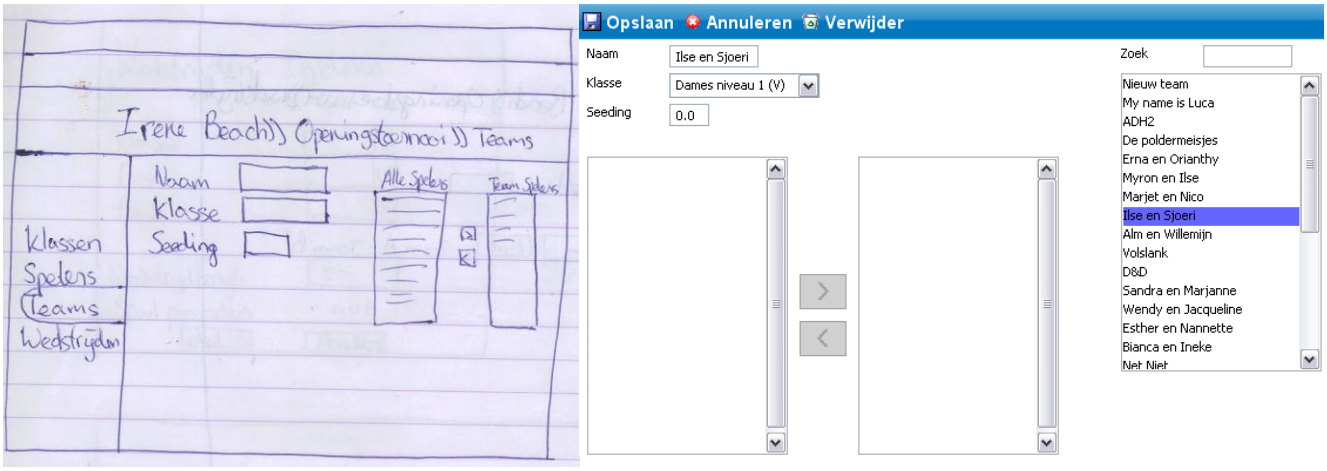
Hier volgen een aantal ruwe schetsen met hun definitieve schermen in de applicatie. Om de definitieve schermen leesbaar te houden wordt één volledig scherm getoond waarna van de andere schermen alleen de dynamische content wordt getoond:



Afbeelding 10: Schermontwerp klassen



Afbeelding 11: Definitieve scherm klassen



Afbeelding 12: Team scherm

De schetsen zijn vooral gemaakt om een globaal beeld te krijgen van het te bouwen scherm met de componenten die daarin nodig zijn. Deze schetsen zijn wel besproken met een paar mensen die een wat technischere achtergrond hebben. De definitieve schermen zijn met meerdere medewerkers besproken maar ook met een aantal wedstrijdleiders. Over het algemeen werden de schermen als duidelijk geacht. Wel zijn er uiteindelijk in overleg een aantal kleine aanpassingen gemaakt om het geheel nog beter te laten functioneren.

Hoewel hier over definitieve schermen wordt gesproken, wordt er een soort van showmodel mee bedoeld. Dat betekent dat er nog geen werkende applicatie bestaat, maar dat er al wel een aantal schermen zijn gebouwd om een algemene indruk te kunnen geven. Zoals in paragraaf 2.3.3 is uitgelegd is hiervoor gekozen om de opdrachtgever een echte indruk te kunnen geven hoe het scherm er exact uit kan zien en om daarnaast alvast kennis te maken met SWT.

Om de kwaliteit van alle schermen te waarborgen zijn er tijdens de bouwfase verschillende momenten geweest waar de gehele applicatie doorgenomen werd om alle puntjes op de i te kunnen zetten. Deze momenten waren redelijk gelijk gespreid over de gehele bouw periode.

2.4 Implementatie

Het bouwen van de applicatie nam de meeste tijd in beslag. Een deel van de tijd die gewonnen werd door te kiezen voor een minimaal technische ontwerp werd in deze fase gebruikt. Door dezelfde keuze moest er ook nog over een aantal oplossingen nagedacht worden.

De prioriteit lag bij het bouwen van een goed werkende applicatie die gemakkelijk te bedienen is. Daarbij moest het op zijn minst mogelijk zijn om toernooien te draaien. Pas als dat mogelijk was zou er gekeken worden naar de implementatie van een SOAP client en een server-client model. Er zou op zijn minst een werkend product opgeleverd worden.

Op het moment van schrijven is er slechts een lokaal werkende applicatie met grafische user interface. Dit zijn dan ook de enige componenten die besproken zullen worden in deze scriptie.

2.4.1 Kern (Model)

Het gehele klassendiagram (Afbeelding 9) kon in één keer overgenomen worden om de basis applicatie te laten werken. Uiteindelijk zijn er een aantal zaken opgesplitst in verschillende klassen om het overzicht te bewaren. Uiteraard moesten een aantal (kleine) aanpassingen plaatsvinden om bepaalde functionaliteit op een eenvoudigere manier aan de praat te krijgen. De meest interessante oplossingen / wijzigingen zullen in dit deel worden behandeld.

2.4.1.1 Wedstrijdschema's

Voor de wedstrijdschema's was het de taak om alle typen schema's op eenzelfde manier te kunnen verwerken. Zoals het vooronderzoek heeft uitgewezen word er binnen Beachvolleybal alleen met Poule en Elimination schema's gewerkt. De pouleschema's zijn eenvoudig door een algoritme te genereren en om een dergelijk schema uit een database tabel te laden heeft men alleen de velden nodig van de beide teams die tegen elkaar moeten spelen.

Een Single- of Double Elimination systeem werkt wat ingewikkelder. Daar is het niet mogelijk om simpelweg elk team tegen elkaar laten spelen, maar moeten er koppels gemaakt worden. Wanneer de koppels bekend zijn moeten alle vervolgwedstrijden, met de juiste teams, op een eenvoudige manier aangemaakt kunnen worden. Ook het correct weergeven van dergelijke schema's is een behoorlijke klus. Daar ga ik dieper op in bij het deel over het bouwen van de GUI.

De simpelste oplossing voor het verwerken van eliminatie schema's was het gebruiken van één tabel met alle gegevens erin. Hierbij zou elk team een uniek cijfer (id) krijgen en zou elk team een ander team treffen in de eerste ronde. Hierbij bestaat een wedstrijd uit twee velden, namelijk de id's van de teams die tegen elkaar spelen. Voor diezelfde wedstrijd worden twee velden toegevoegd die ook uit een uniek cijfer bestaan. De extra velden staan voor het id wat het winnende en verliezende team krijgt. Deze id's kunnen in opvolgende wedstrijden gebruikt worden.

Hier volgt een voorbeeld van een Double Elimination met acht teams:

	Thuis	Uit		Winnaar	Verliezer		speelronde
1	1	8		10	21		1
2	4	5		11	22		1
3	3	6		12	23		1
4	2	7		13	24		1
5	10	11		14	31		2
6	12	13		15	32		2
7	23	24		25			2
8	21	22		26			2
9	14	15		16	41		3
10	31	26		27			3
11	32	25		28			3
12	27	28		17			4
13	41	17		18			5
14	16	18					6

Tabel 1: Dubbele eliminatie

In Tabel 1 is te zien dat er met acht teams word gestart. Deze zijn voor het gemak 1 tot en met 8 genummerd.

Wanneer Team 1 zijn eerste wedstrijd wint krijgt het team nummer 10.

Wint nummer 5 ook zijn eerste wedstrijd dan krijgt dat team nummer 11.

In wedstrijd 5 is te zien dat nummer 10 en 11 tegen elkaar spelen wat er op neerkomt dat dit Team 1 tegen Team 5 is.

Dit concept gaat net zolang door tot er geen winnaars of verliezers meer zijn vastgesteld. Uiteindelijk komt het uit op een finale tussen twee teams. Dit schema is door de applicatie eenvoudig te interpreteren door de teamid's van de wedstrijd terug te leiden naar het originele team. Wil men bijvoorbeeld weten wie er in wedstrijd zeven tegen elkaar spelen dan herleid de applicatie de nummers 23 en 24 terug naar wedstrijd drie en vier. Indien wedstrijd drie door team drie is gewonnen dan laat de applicatie dit ook zien. Is wedstrijd drie nog niet gespeeld dan kan de applicatie de teamnaam eenvoudig als L 3 (Loser 3) weergeven.
Hoe de applicatie dit visueel weergeeft is te zien in Afbeelding 4. In die afbeelding zijn nog geen wedstrijden gespeeld.

Een dergelijk schema is ook eenvoudig toe te passen voor alleen Single Elimination door enkel de winnende teams een nieuw id te geven. Ook een simpel poule systeem is mogelijk door alleen het thuis en uit team in te vullen. Overigens is een poule systeem ook eenvoudig te genereren door de applicatie.

2.4.1.2 Wedstrijden

In eerste instantie was het de bedoeling dat elke wedstrijd een referentie had naar beide Team objecten om te weten welke teams er tegen elkaar zouden spelen. Iets dat redelijk eenvoudig te implementeren was. Dit onderdeel deed dan ook zijn werk zonder problemen.

De problemen ontstonden bij het testen van wat toernooitjes. Stel ik organiseer een toernooi en heb alles binnen de applicatie voorbereid. Alle klassen, met bijbehorende fasen en poules zijn aangemaakt. De teams zijn vervolgens aan deze klassen toegewezen. De wedstrijdschema's zijn ingesteld en ik heb de juiste tijd aan elke wedstrijd toegewezen. Kortom, ik ben helemaal klaar voor dat toernooi.

Dan krijg ik opeens een telefoontje. Team A wil ook nog graag meedoen met het toernooi. Kan dat? Uiteraard kan dat, daar hebben we immers een flexibel systeem voor. Dus ik voeg Team A toe aan klasse B en klik op opslaan. Vervolgens ga ik naar mijn wedstrijdschema kijken en zie daar alle wedstrijd objecten zijn opnieuw aangemaakt en daarmee zijn dus alle data en tijden van de te spelen wedstrijden (in klasse B) verloren gegaan.

Het beoogde systeem werkte dus perfect tot er bepaalde wijzigingen werden aangebracht. Deze wijzigingen hadden zulke dramatische gevolgen dat er op dit gebied wel iets moest veranderen. Wat er namelijk gebeurd is dat het systeem bij het wijzigen van de poule-indeling (teams op een andere positie zetten, teams toevoegen of verwijderen) niet meer weet welke wedstrijden er nog kloppen.

Op zich is het mogelijk om uit te zoeken welke wedstrijden verwijderd moeten worden of van welke wedstrijden andere teams moeten worden ingesteld. Toch zou het algoritme makkelijker kunnen wat tot een kleine, maar ingrijpende, wijziging zou moeten leiden.

De oplossing ligt in het niet hard vastleggen van de teams per wedstrijd, maar in het verwijzen naar de positie van het team in de poule. Deze positie komt vervolgens overeen met een positie in de array met teams binnen de poule. Als vervolgens Team A van positie 5 naar positie 3 in de poule wordt verplaatst, zal dit automatisch opgemerkt worden door de wedstrijd objecten.

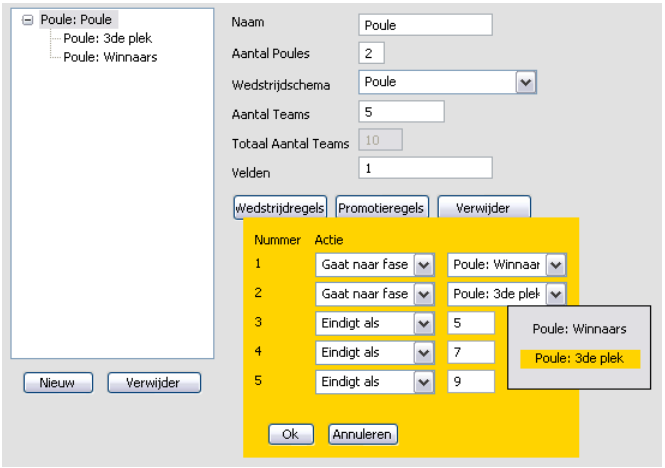
Deze wijziging had uiteindelijk meer gunstige effecten, met name bij elimination schema's. Voordat deze wijziging werd aangebracht moesten er voor alle vervolgwedstrijden (kwartfinales, halve finales etc.) placeholder objecten gemaakt worden voor de teams. Het was namelijk nog niet bekend welke teams in de finales terecht zouden komen.
Na deze wijziging was het niet meer noodzakelijk om daar placeholder objecten aan toe te wijzen omdat het simpel was om bepaalde posities terug te leiden naar het uiteindelijke team zoals uitgelegd in deel 2.4.1.1.

2.4.1.3 Vervolgfasen

Het komt geregeld voor dat er in een toernooi eerst een poule fase en daarna een eliminatie fase gespeeld wordt. Daarnaast is het ook denkbaar dat er eerst in vijf poules gespeeld zal worden en dat daarna de nummers één elkaar treffen in een winnaars poule en dat de nummers twee elkaar treffen in een poule etc. Zo komen er binnen een toernooi nog vele andere combinaties voor waar rekening mee gehouden moet worden.

Om alle mogelijke combinaties ook daadwerkelijk te realiseren is een klasse gemaakt die wat werk overneemt van de Fase. Deze klasse, PromotieRegels genaamd, is eigenlijk een veredelde lijst met items. Hij bevat, afhankelijk van hoe de gebruiker het instelt, 0 tot het maximale aantal teams binnen een poule aan items die kunnen vertellen waar een team aan het eind van de fase naartoe moet. Deze items zijn gelinkt aan de eindstand van de poule. Zo is dus in te stellen dat alle nummers 1 van een poule naar de winnaarsfase gaan. En voor de nummers 2 is in te stellen dat ze gezamenlijk een 5de plek krijgen in de eindstand van de klasse.

In Afbeelding 13 is het scherm om fasen van een klasse te bewerken te zien. Het linker menu geeft de mogelijkheid om oneindig veel nieuwe fasen aan de geselecteerde fase te hangen. De fase die daar is geselecteerd is de eerste fase van de klasse die Poulefase is genoemd. In die fase zitten twee poules met elk vijf teams. Wanneer er op de knop 'Promotieregels' wordt gedrukt zal er een extra scherm verschijnen met de opties voor de nummers 1 tot en met 5 van de poules.



Afbeelding 13: Scherm Promotieregels

In dat scherm is te zien dat de nummers 1 van beide poules naar de winnaarspoule zullen gaan en dat de nummers 2 van beide poules om de derde plek zullen strijden. De nummers 3 tot en met 5 zullen na de poulefase klaar zijn en als gedeelde 5de, 7de en 9de in de eindstand van de klasse verschijnen.

2.4.2 Grafische User Interface

Op dit moment zijn er drie noemenswaardige GUI toolkits beschikbaar voor Java: AWT (Abstract Window Toolkit), Swing en SWT (Standard Widget Toolkit). De eerste twee zijn door Sun ontwikkeld en worden daarom standaard met Java meegeleverd. SWT is door IBM ontwikkeld en dient als aparte library gebruikt te worden.

Met AWT en Swing is de nodige ervaring opgebouwd. Daarentegen is er geen ervaring met SWT. Aangezien er lovende verhalen over SWT worden verteld was het wel de moeite waard om daar een klein onderzoekje naar uit te voeren.

2.4.2.1 De verhalen over SWT

Verschillende mensen zijn van mening dat SWT een zeer goed alternatief is voor AWT en Swing. SWT zou een krachtige toolkit zijn die goede prestaties levert en volledig is geïntegreerd in het besturingssysteem. Daarmee zou SWT mooi en snel aanvoelen in tegenstelling tot het 'lelijke' en trage Swing. Mooi en lelijk is een kwestie van smaak. Wat hier voornamelijk mee bedoelt wordt is dat een knop in SWT er ook echt uitziet als een Windows knop (als de applicatie op Windows draait) terwijl een knop in Swing er uit probeert te zien als een Windows knop.

Deze verhalen klinken veelbelovend omdat gebruikers zitten te wachten op (voor hen) bekende componenten die natuurlijk aanvoelen. Een applicatie die identieke componenten heeft als een willekeurige andere Windows applicatie zal dus sneller geaccepteerd worden.

2.4.2.2 Swing versus SWT

In het begin leek AWT nog noemenswaardig. Toch is deze niet vergeleken met Swing en SWT omdat AWT te gedateerd is. AWT ondersteunt te weinig componenten om daar een complexe applicatie mee te bouwen. Daarnaast wordt Swing als opvolger van AWT gezien (Swing moest de vele problemen van AWT oplossen) en daarmee geeft Sun aan dat men Swing in plaats van AWT zou moeten gebruiken.

Swing

Swing is ontwikkeld om de nadelen van AWT op te lossen. De nadelen van AWT zijn vooral de magere set componenten die beschikbaar zijn en de afhankelijkheid van het besturingssysteem.

Om dit doel te realiseren zijn alle grafische componenten volledig in Java code geschreven. Deze componenten dienen daardoor ook door Java zelf getekend te worden. Het voordeel hiervan is dat er inderdaad zoveel mogelijk componenten ontwikkeld kunnen worden en dat ze er op alle besturingssystemen exact hetzelfde uit zullen zien. Het nadeel is dat het component nooit exact hetzelfde eruit kan zien als een soortgelijk native component. Een button met Swing ziet er dus niet hetzelfde uit als een button in een native Windows applicatie.

Een ander nadeel van Swing is dat het geen ondersteuning heeft voor hardwareversnelling. Dat betekent dat het tekenen van een component in Swing over het algemeen langer duurt dan het tekenen van een component in AWT of SWT. Hoewel Sun hard aan de prestaties van Swing heeft gewerkt zal er altijd een krachtigere computer nodig zijn voor een Swing applicatie.

Niet alle nadelen van Swing hoeven van toepassing te zijn. Voor een applicatie hoeft het niet uit te maken of het component precies op zijn gelijke native component moet lijken. Wanneer men de componenten er helemaal op zijn eigen manier uit wilt laten zien gaat dit nadeel helemaal niet op.

Overigens heeft Sun uitstekende documentatie voor Swing gemaakt. Het bouwen van een Swing applicatie is zeer eenvoudig door de vele tutorials en voorbeelden. Daarnaast is de javadoc zeer uitgebreid en wordt er, indien er interesse voor is, ook nog wat technische achtergrondkennis gegeven.

SWT

In tegenstelling tot het gevestigde Swing is SWT een opkomende toolkit die nog volledig in ontwikkeling is. Op het moment van schrijven dateert de laatste stable release (3.5.2) uit februari 2010. SWT word steeds vaker gebruikt voor het bouwen van applicaties. Onder andere Eclipse, IBM Rational Software, IBM Lotus Software en Vuze (voorheen Azureus) zijn geschreven in SWT. Iets dat aantoont dat SWT inmiddels een volwassen toolkit is.

SWT volgt de principes van AWT door gebruik te maken van native libraries. Waar AWT slechts de componenten gebruikt die in alle besturingssystemen voorkomen heeft SWT extra native libraries ontwikkeld voor de componenten die niet ondersteund worden. Daarmee hebben SWT applicaties niet alleen dezelfde look & feel als het besturingssysteem maar ook de prestaties daarvan.

De set componenten die SWT kan aanbieden is op deze manier grotendeels gelijk aan de set componenten die Swing kan aanbieden. Swing biedt echter verreweg meer opties om de componenten te manipuleren. Zo is het met Swing wel mogelijk om een Button een andere achtergrond te geven.
Aangezien de bedoeling van SWT in eerste instantie is om juist het gevoel te geven dat men met een lokale applicatie aan het werk is, is het nog maar de vraag hoe belangrijk het is om de componenten er anders uit te laten zien dan wat standaard is in het besturingssysteem. Mocht een ontwikkelaar toch voorkeur hebben voor een andere weergave, zoals een simpele knop, dan kan men alleen gebruik maken van een container om vervolgens zelf de volledige component op het scherm te tekenen of uit andere componenten op te bouwen.

Een groot nadeel van SWT is dat het moeilijk is om goede en duidelijke documentatie te vinden. Daar waar Swing voor elk component uitgebreide documentatie heeft met alle details en mogelijkheden moet SWT het slechts met (magere) voorbeelden en tutorials doen.

Ook maakt SWT niet altijd waar wat het belooft. Zo laten een aantal benchmarks[2] zien dat SWT niet altijd sneller is dan Swing. Alleen op Windows is SWT duidelijk sneller dan Swing. Op andere besturingssystemen is het nog maar de vraag of SWT zijn woorden wel waar kan maken.

Alles op een rijtje

Om een duidelijk overzicht van alle voor- en nadelen van elke toolkit te krijgen zijn deze in een tabel verwerkt.

	Swing	SWT
Integratie in besturingssysteem	Nee	Ja
Aantal componenten	++	++
Features	++	o
Prestaties	+	++
Documentatie	++	-

Tabel 2: Vergelijking tussen Swing en SWT

Integratie in het besturingssysteem doelt op het gebruik van lokale componenten waardoor iemand het gevoel zal hebben dat hij echt met een applicatie werkt die voor het eigen besturingssysteem is gemaakt.
Het aantal componenten doelt op het aantal componenten die standaard beschikbaar zijn bij de toolkit.
De features doelt op de mogelijkheden die er zijn om de component op een bepaalde manier te presenteren.
De prestaties heeft voornamelijk te maken met de snelheid en reactietijd van de interface.
De documentatie gaat over de beschikbare documentatie en de kwaliteit daarvan.

Swing of SWT?

Swing en SWT hebben allebei hun voor- en nadelen. Er kan gesteld worden dat de voordelen van beide toolkits de oplossing van de nadelen van de andere toolkit zijn. Zo levert SWT de prestaties en de native look en feel, maar levert Swing een uitgebreide featureset en platform onafhankelijkheid.

Swing lijkt een goede optie omdat Swing uitstekende documentatie heeft en de platform onafhankelijkheid biedt. Dat laatste komt goed uit omdat de MacBook ook ondersteund moet worden.

Toch kies is gekozen voor SWT en dat met name met het oog op de toekomst en gebruikservaring. Gebruikers neigen om sneller applicaties te gebruiken die er bekend uitzien

[2] Bron: Križnar, Igor (2005), SWT Vs. Swing Performance Comparison. Geraadpleegd op 6 mei 2010, http://cosylib.cosylab.com/pub/CSS/DOC-SWT_Vs._Swing_Performance_Comparison.pdf

en redelijk vlot reageren op hun acties. Daarnaast is SWT nog volop in ontwikkeling om het leven in de toekomst alleen maar mooier te maken!

Er komen wel een aantal risico's bij het gebruik van SWT kijken. Zo kan het project vertraging oplopen door de weinige kennis over SWT. Daarnaast dient de applicatie op verschillende besturingssystemen (vooral op Mac OS) getest te worden voordat de applicatie ook daadwerkelijk voor deze systemen gebruikt kan worden. Dit zou nog de nodige problemen kunnen veroorzaken.

2.4.2.3 Bouwen met SWT

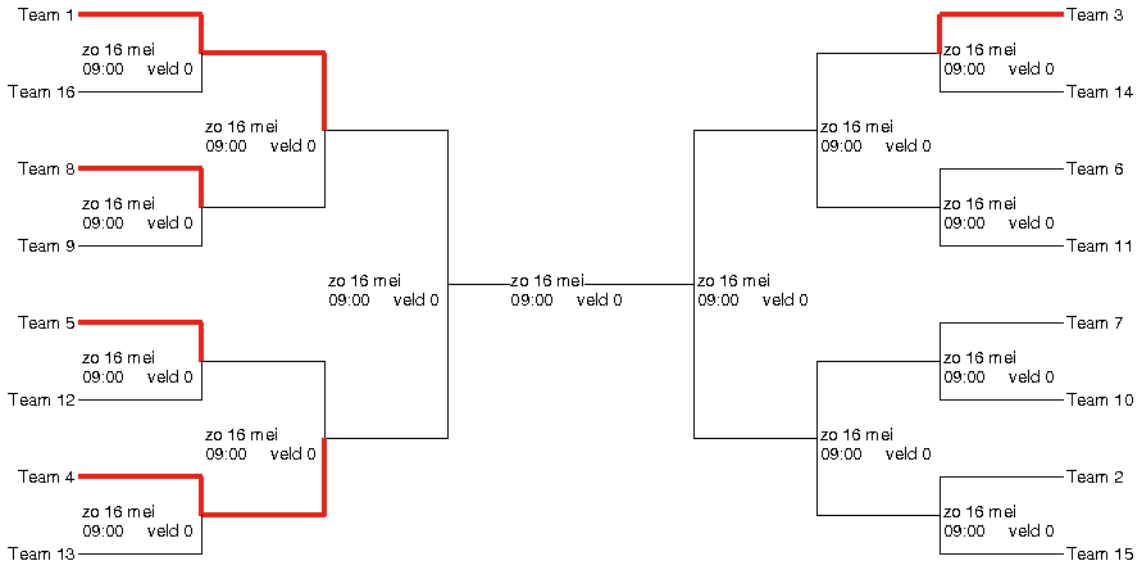
Wanneer er eenmaal ervaring met SWT is opgedaan, is het bouwen van schermen redelijk eenvoudig. De kern van de GUI bestaat uit het globale scherm, het scherm zonder dynamische content, en een aantal abstracte klassen. Deze abstracte klassen zorgen voor de implementatie van alle mogelijke schermen. Zo is er de klasse ContentScreen voor gewone schermen waar niets bijzonders in gebeurd, maar is er ook een klasse FormScreen die automatisch een toolbar met de knoppen opslaan/annuleren/verwijderen toont. Het FormScreen implementeert daarmee ook de abstracte methoden save(), cancel() en delete() om ervoor te zorgen dat deze knoppen ook bruikbaar zijn.
Wat ook tot de kern behoord is het zogenoemde CrashScreen. Dit scherm wordt getoond wanneer er een Exception optreedt die niet door de GUI wordt afgevangen. Hierdoor is het onmogelijk dat de applicatie door bijvoorbeeld een NullPointerException zomaar afsluit waarmee alle gegevens verloren kunnen gaan. Uiteraard blijft het streven om een dergelijk scherm zo min mogelijk te laten tonen.

Color, Font & Image

In tegenstelling tot Swing beheert SWT geen zelf aangemaakte componenten. Als een Button wordt aanmaakt moet er ook voor gezorgd worden dat deze uit het geheugen wordt verwijderd. Gebeurt dat niet? Dan bereikt de applicatie ooit het maximum aantal objecten dat een besturingssysteem kan verwerken, met een crash als gevolg.
Voor componenten is dat in deze applicatie geen probleem aangezien de applicatie alle componenten van het oude scherm uit het geheugen verwijderd tijdens het aanroepen van een nieuw scherm. Waar wel opgepast mee moet worden is het gebruik van Color, Font & Image. Deze worden namelijk niet samen met de component uit het geheugen verwijderd en daarmee blijven oude instanties altijd bestaan. Door hier een eigen 'manager' voor te ontwikkelen moeten dergelijke problemen voorkomen worden.
Deze manager heeft de methoden getFont, getColor en getImage die allen de parameters vereisen om het gewenste object te krijgen. Vervolgens bekijkt de manager of een dergelijk object al aanwezig is binnen het systeem en geeft dat object terug. Wanneer een dergelijk object nog niet aanwezig is wordt deze aangemaakt en binnen het register opgeslagen.

2.4.2.4 Wedstrijdschema's en Standen

Het bouwen van een grafische user interface is op zichzelf niet heel spannend. Wanneer men de toolkit eenmaal onder zijn knie heeft is het slechts een kwestie van componenten op het scherm toveren. Dit kan zelfs als saai werk worden beschouwd omdat dit in de meeste gevallen echt simpel typewerk is. Een enkele keer komt het voor dat er een wat interessanter scherm gebouwd moet worden en ook bij dit project is dat het geval. Namelijk het tonen van de Single- en Double elimination wedstrijdschema's en standen.



Afbeelding 14: Single Elimination

Het eindresultaat is te zien in Afbeelding 14. In deze afbeelding is een schema te zien waar duidelijk uit blijkt welke teams tegen elkaar zouden moeten spelen en welk team ze zullen treffen wanneer ze winnen.

Om dit technisch te realiseren is de vergelijking met een boom (Tree) gemaakt. Deze begint bij de finale die twee takken heeft, de halve finales. De halve finales hebben ook weer twee takken, de kwartfinales. De kwartfinales hebben vervolgens ook weer twee takken die voor de 8ste finales staan. Dat zijn tevens de eerste wedstrijden in dit schema en dus stopt de boom op die plek.

Het tekenen van dit schema begint bij de laatste wedstrijd, de finale. Dit is enkel een lijn in het midden van het scherm. Daarna worden twee methodes aangeroepen: drawLeft om de volgende wedstrijden links van de finale te tekenen; drawRight om de volgende wedstrijden rechts van de finale te tekenen.

Deze methoden hoeven alleen te weten vanaf welk coördinaat ze moeten beginnen met tekenen, welke ronde er getekend wordt (halve finale, kwartfinale, etc.) en wat het maximum aantal rondes is in dat schema. Aan de hand van de huidige ronde en het maximum aantal rondes is uit te rekenen hoe hoog de tak moet worden. Per ronde groeit de hoogte exponentieel (hij verdubbeld).

Stel dat de takken in de eerste ronde 50 pixels hoog zullen zijn. Dan moeten ze in de tweede ronde 100 pixels zijn, in de derde ronde 200 pixels en in de vierde ronde 400 pixels. Afhankelijk van het aantal rondes kan dat dus oneindig lang doorgaan.

Wanneer de hoogte bekend is, is het slechts een kwestie om vanaf het startpunt een lijntje van de halve hoogte omhoog en omlaag te tekenen. Dan worden er aan de uiteinden twee lijntjes horizontaal getekend en kan de volgende drawLeft of drawRight methode worden aangeroepen met het nieuwe startpunt en de ronde die met één is verhoogd.

Omdat voor tekenvlakken altijd een hoogte en breedte nodig is dienen deze ook uitgerekend te worden. Dat is eenvoudig te doen met behulp van $2^{<\text{aantal rondes}>}$. De hoogte is vervolgens de hoogte van de eerste tak maal $2^{<\text{aantal rondes}>}$. De breedte is de breedte van de eerste tak maal $2^{<\text{aantal rondes}>}$.

2.4.3 SOAP Client

Op het moment van schrijven is de SOAP Client nog niet gerealiseerd. De oorzaak hiervan ligt bij het feit dat er ook andere zaken aan de website gewijzigd moesten worden die prioriteit hadden boven de applicatie. Een WSDL bestand is wel opgesteld, maar daar doen zich nog een aantal problemen mee voor. Het is dus nog niet mogelijk om een werkende SOAP Client te realiseren.
De externe ontwikkelaar heeft naar eigen zeggen begin juni tijd om hiernaar te kijken. Het is dus wel de bedoeling dat een dergelijke client nog binnen het project wordt gerealiseerd.

2.4.4 Lokaal netwerk

Ook het lokale netwerk is nog niet gerealiseerd. Dit component heeft de laagste prioriteit omdat er maar een paar toernooien worden georganiseerd die een dergelijke situatie vereisen. De vraag is of het realiseren van dit component ook gaat lukken binnen het project. Het is belangrijker om een goed functionerende applicatie te hebben met minder functionaliteit. Ook het realiseren van een SOAP Client is belangrijker en zal zeker eerst gebeuren. Ook daar komt nog het nodige testwerk bij. Indien alles voorspoedig loopt zal hier nog naar gekeken worden. Maar op dit moment is de verwachting dat dit component niet voor 30 juni geïmplementeerd zal worden.

2.5 Testen

Op het moment van schrijven is er nog niet veel getest. Dit komt doordat het tot kort geleden niet mogelijk was om een volledig toernooi te kunnen draaien. Daarnaast begonnen de eerste toernooien waar getest kon worden pas op half mei. Wanneer alle functionaliteit, exclusief de SOAP Client en het lokale netwerkgedeelte, is geïmplementeerd kan uitgebreider getest gaan worden. De verwachting is dat dit in de tweede week van juni plaats kan vinden.
Testen heeft prioriteit boven de SOAP Client en het netwerkcomponent omdat de applicatie als zichzelf goed zijn werk moet kunnen doen. Als eerst een SOAP Client wordt geïmplementeerd bestaat het risico dat er nog de nodige fouten in het geheel zitten wat de gehele applicatie onbruikbaar zou kunnen maken.

Zolang het project loopt zal de meeste testtijd gaan zitten in het draaien van toernooien in de applicatie. Het project loopt in principe tot eind juni en tot die tijd wordt er elke week een toernooi georganiseerd. Deze toernooien moeten de belangrijkste elementen zoals het invoeren van teams, poule-indeling, wedstrijdschema's en verwerken van uitslagen testen.

Voor minder voorkomende acties zullen een aantal testcases gemaakt moeten worden. Dit betreft voornamelijk het beheren van verschillende fasen per klasse. Ook het automatisch indelen van de wedstrijden zal het nodige testwerk vereisen.

2.6 Implementatieplan

Het implementeren van een Java applicatie is op zichzelf niet zo ingewikkeld. De gebruiker heeft immers enkel de Java Runtime Environment nodig om vervolgens de applicatie te kunnen starten.

Gebruiksvriendelijkheid staat echter voorop en men kan niet van de gebruikers verlangen om de applicatie via de command-line te starten. Een executable jar bestand werkt niet op elke computer en word door de gebruikers ook niet altijd als een applicatie aangezien.
Er zal dan ook gebruik gemaakt moeten worden van Java wrappers om de gebruikers het idee te geven dat ze een 'echte' applicatie starten. Voornamelijk voor Mac OS en Windows is dit belangrijk. Linux gebruikers hebben doorgaans meer verstand van zaken en krijgen daarbij een lagere prioriteit.

De applicatie zal als download aangeboden worden op de website beachcompetitie.nl. Dit zal ook meteen het gebruik van de website bevorderen. Alle beach- en zaalvolleybalverenigingen zullen een mail ontvangen met het nieuws dat deze applicatie is ontwikkeld en dat deze vrij beschikbaar is op het internet. Bijkomend voordeel is dat de toernooi kalender van de Nevobo hierdoor automatisch word aangevuld met extra toernooien omdat de verwachting is dat er ook een aantal verenigingen met de applicatie aan de slag zullen gaan.

Wedstrijdleiders van de nationale competitie zullen tevens een mail ontvangen en daarnaast zal de applicatie op alle toernooilaptops van de Nevobo geïnstalleerd worden.

3

Scriptie

Deel 3: Evaluatie

3.1 Opgeleverde producten

In het plan van aanpak is een lijst met producten opgesteld die opgeleverd dienen te worden. Deze worden in dit deel stuk voor stuk behandeld.

Plan van Aanpak

Status: Opgeleverd

Het plan van aanpak is geheel binnen de geplande tijd opgeleverd.

Technisch Functioneel Ontwerp

Status: Deels opgeleverd

Het technisch functioneel ontwerp is grotendeels opgeleverd binnen de gestelde tijd. De sequence diagrammen zijn uiteindelijk achterwege gelaten daar een WSDL bestand voldeed voor de ontwikkeling van de applicatie.

De applicatie

Status: Deels opgeleverd

Op het moment van schrijven zijn de belangrijkste elementen van de applicatie ontwikkeld. Het is mogelijk om een redelijk complex toernooi te draaien en er zijn ook bepaalde extra's ingebouwd die het organiseren van een toernooi gemakkelijker maken binnen de applicatie. Er resten nog een paar functies die ontwikkeld moeten worden. Echter zijn deze niet cruciaal voor de toernooi-organisatie en er wordt verwacht dat deze binnen het project te realiseren zijn.

De enige componenten die op dit moment niet functioneren zijn de SOAP Client en het lokale netwerk component. Daarvan word verwacht dat de SOAP Client nog binnen het project ontwikkeld kan worden en dat het lokale netwerk een uitbreiding voor de toekomst is.

Documentatie

Status: Nog niet opgeleverd

Op het moment van schrijven is de documentatie nog niet opgeleverd. Deze moet bestaan uit javadoc en een handleiding. Er is een begin gemaakt aan de javadoc echter is deze nog niet ver gevorderd. De handleiding moet nog even op zich laten wachten. Het doel is wel om deze binnen het project op te leveren.

Adviesrapport implementatie

Status: Opgeleverd

In het implementatieplan zijn specifieke punten benoemd en worden er ook andere zaken als het vullen van een toernooikalender bij betrokken.

3.2 Conclusies en aanbevelingen

Het is gelukt om alle hoofd- en deelvragen te beantwoorden en om daar een mooi product uit te ontwikkelen.

3.2.1 Deelvragen

3.2.1.1 Hoe wordt een toernooi/competitie georganiseerd?

Zoals beschreven in paragraaf 2.2.1.1 wordt elk toernooi volgens een vast proces georganiseerd. Globaal houdt dit proces in dat de faciliteiten worden vastgesteld (locatie, aantal velden etc.) waarna de inschrijvingen kunnen worden geopend. Wanneer de inschrijfperiode is verlopen worden de inschrijvingen gesloten en kan het echte werk beginnen. Na de inschrijvingen moeten de poule indelingen en wedstrijdschema's worden gemaakt. Daarna staat er niets meer in de weg om te kunnen volleyballen.

3.2.1.2 Waar zitten de ergernissen bij het organiseren van een toernooi?

Van ergernissen is niet echt te spreken. Wel is het zo dat er een aantal zaken een stuk makkelijker gemaakt kunnen worden voor organisatoren/wedstrijdleiders. Het proces omvat een aantal tijdrovende zaken zoals de poule indeling (paragraaf 2.2.1.3) en het creëren van wedstrijdschema's (paragraaf 2.2.1.4). Tijdens een toernooi kan het verwerken van uitslagen en het berekenen van standen nogal hectisch verlopen door de snelheid waarmee dit soms moet worden afgehandeld (paragraaf 2.2.1.5).

3.2.1.3 Wat kan de website beachcompetitie.nl voor een geautomatiseerd systeem betekenen?

De website beachcompetitie.nl bestaat uit een inschrijfsysteem voor beachvolleybal toernooien. Dat systeem omvat ook een toernooikalender en de nationale rankings van de teams die meedoen met de nationale competitie. Zie voor meer informatie Bijlage 1.

De ontwikkelaar van deze website zal webservices gaan implementeren om communicatie tussen de applicatie en het systeem mogelijk te maken. Deze services zullen het voor gebruikers gemakkelijker maken de inschrijvingen in de applicatie te verwerken. Daarnaast kan de applicatie eenvoudig uitslagen en standen op het internet publiceren.

3.2.1.4 Wat voor koppeling dient er met de website beachcompetitie.nl te zijn?

Zoals in paragraaf 2.3.1.1 beschreven wordt er gebruik gemaakt van SOAP webservices om te communiceren met de website. Hiervoor is een WSDL bestand opgesteld door de ontwikkelaar zodat er een SOAP Client gerealiseerd kan worden.

3.2.1.5 Aan welke technische (infrastructuur) eisen moet de automatisering voldoen?

Zoals beschreven in paragraaf 2.2.2 zijn er evenementen waar gebruik gemaakt wordt van aggregaten waardoor er rekening gehouden dient te worden met stroomuitval, wat uiteraard gepaard gaat met netwerkuitval.
De applicatie zal dan ook regelmatig backups moeten maken en gedurende de perioden dat er wel netwerkverbinding is na elke actie synchroniseren met de andere clients. Daarnaast dienen de gegevens van het inschrijfsysteem offline beschikbaar te zijn.

In dezelfde paragraaf wordt ook verteld dat de applicatie platform onafhankelijk dient te zijn omdat er, naast de Windows gebruikers, ook eindgebruikers zullen zijn die beschikken over een MacBook

3.2.1.6 Wat zijn de beste technieken voor het realiseren van een geautomatiseerd systeem?

Omdat de applicatie platform onafhankelijk dient te zijn kan het bijna als vanzelfsprekend worden genoemd dat deze in Java wordt geschreven. Voor de componenten binnen de applicatie is het antwoord echter wat moeilijker te geven.
In Afbeelding 8 (Architectuur applicatie) is te zien dat de applicatie te maken heeft met de GUI, SOAP Client, Server en Client.

Swing en SWT zijn beiden krachtige GUI toolkits voor Java applicaties. Swing heeft vooral een zeer uitgebreide featureset en doet zijn werk goed op meerdere platformen. Toch is SWT in het voordeel omdat deze toolkit een betere gebruikservaring kan leveren. Daarnaast is SWT nog volop in ontwikkeling waardoor deze toolkit in de toekomst nog beter zal functioneren.

Voor de communicatie tussen het inschrijfsysteem en de applicatie zijn webservices de grote winnaar. Webservices zijn juist gemaakt als oplossing voor dergelijke communicatie en er hoefde dan ook niet getwijfeld te worden over andere oplossingen.

Het is nog niet duidelijk wat de beste techniek is voor de server-client oplossing voor het lokale netwerk. Andere zaken die meer tijd kostten dan gepland hadden prioriteit boven het realiseren van het lokale netwerk component. Wel is het duidelijk dat de meest geschikte technieken RMI of Sockets zijn.

3.2.2 Hoofdvraag

Aan de hand van deze deelvragen kan een antwoord gegeven worden op de hoofdvraag **"Hoe kan de Nevobo het organiseren van toernooien en competities automatiseren zodat er veel tijd bespaard blijft met het maken van wedstrijdschema's en er een koppeling ontstaat met de website beachcompetitie.nl voor het automatisch uitwisselen van centraal opgeslagen gegevens."**

Er dient een applicatie gebouwd te worden die het volledige toernooi organisatie proces kan doorlopen en daarmee kan ondersteunen tijdens de arbeidsintensieve stappen. Dit betreft vooral het maken van een poule indeling en wedstrijdschema's. Daarnaast ondersteunt deze applicatie ook bij het verwerken van uitslagen en het tonen van standen binnen een poule.

Tijdens de ontwikkeling wordt er rekening gehouden met platform onafhankelijkheid en eventuele afwezige faciliteiten zoals internet.

Door webservices te gebruiken om gegevens uit te wisselen met de website beachcompetitie.nl (het inschrijfsysteem) kan ook het verwerken van de inschrijvingen sterk vereenvoudigd worden. Tevens biedt dit de mogelijkheid tot het online publiceren van uitslagen en standen.

3.2.3 Aanbevelingen

Het was niet mogelijk om alle componenten binnen het project te realiseren. Zo is het onwaarschijnlijk dat het na dit project mogelijk zal zijn om meerdere applicaties binnen een lokaal netwerk te laten functioneren.

Het is verstandig om eerst uit te zoeken wat de absolute noodzaak is van het gebruiken van meerdere computers tijdens een evenement. Indien meerdere computers noodzakelijk zijn in verband met een grote afstand tussen verschillende wedstrijdleiders kan er ook gekeken worden naar andere communicatiemiddelen. Het versturen van uitslagen zou bijvoorbeeld ook via de mail kunnen of het versturen van een bestand via het netwerk zodat deze in de applicatie verwerkt kunnen worden.

Omdat de applicatie redelijk veel functionaliteit bevat kan gekeken worden of deze misschien ook te gebruiken is voor zaalvolleybal. Als dat niet het geval is, is de vraag hoeveel werk het zou zijn om extra functionaliteit te implementeren zodat de applicatie ook voor zaalvolleybal gebruikt kan worden. Voor zaalvolleybal bestaat namelijk nog geen toernooi applicatie en ook daar worden toernooien voor georganiseerd.

Tenslotte kan er gekeken worden hoe eenvoudig het is om de applicatie met extra functionaliteit uit te breiden. Hierbij wordt gedoeld op zaken als scheidsrechters-administratie en ledenadministratie etc.

3.3 Evaluatie

Over het algemeen verliep het project voorspoedig. Na een paar weken tijd kon er begonnen worden met bouwen. Goede communicatie met wedstrijdleiders in het begin heeft eraan bijgedragen dat er een duidelijk klassendiagram opgesteld kon worden en dat er tijdens het bouwen niet opeens van alles omgegooid hoefde te worden. Uiteraard zijn er een paar uitzonderingen zoals het verwijderen van de directe referenties tussen de wedstrijden en de teams. Deze wijzigingen hebben echter niet voor veel vertraging gezorgd.

Wat wel opviel is dat het behoorlijk veel tijd kost om een goede user interface te bouwen. Het bouwen in SWT zelf is niet ingewikkeld maar om overal rekening mee te houden binnen een scherm is best lastig. Zaken als het eenvoudig kiezen van teams binnen een klasse uit een lijst neemt gewoon veel tijd in beslag. Evenals het bouwen van een popupscherm en het mogelijk maken om snel en eenvoudig te wisselen tussen verschillende schermen etc. kost tijd. Het zal opvallen als deze onderdelen ontbreken. Het bouwen van een scherm is dus niet zo lastig maar alle functionaliteit toevoegen om het scherm eenvoudig te bedienen wel.

Omdat het bouwen van de user interface meer tijd in beslag nam ging dat ten koste van het bouwen van andere zaken zoals de server en de client. Dat is dan ook een component dat nog niet gerealiseerd is en waarschijnlijk ook niet meer gerealiseerd gaat worden binnen het tijdsbestek van dit project. Echter heeft een goed werkende applicatie prioriteit boven een slecht werkende applicatie met meer functionaliteit.

Hoewel er niet veel inhoudelijk contact was met de opdrachtgever is er wel contact geweest met andere medewerkers van de Nevobo. Dit ging vooral over de te gebruiken technologieën, de schermontwerpen en de functionaliteit van de applicatie. De contactmomenten waren goed gespreid over het gehele project.

Hoewel de planning iets te optimistisch is gemaakt en nog niet alle functionaliteit binnen de applicatie aanwezig is, is er wel een goed werkende applicatie uit voortgekomen. De applicatie kan de belangrijkste zaken uitvoeren en daar gaat het uiteindelijk om. Er kan dan ook gezegd worden dat het een geslaagd project is!

Bronvermelding

Barry Feigenbaum (2006), SWT, Swing or AWT: Which is right for you? Geraadpleegd op 5 mei 2010, http://www.ibm.com/developerworks/grid/library/os-swingswt/

Križnar, Igor (2005), SWT Vs. Swing Performance Comparison. Geraadpleegd op 6 mei 2010, http://cosylib.cosylab.com/pub/CSS/DOC-SWT_Vs._Swing_Performance_Comparison.pdf

Krug, S. (2005). Don't Make Me Think! A Common Sense Approach to Web Usability. Berkeley: New Riders.

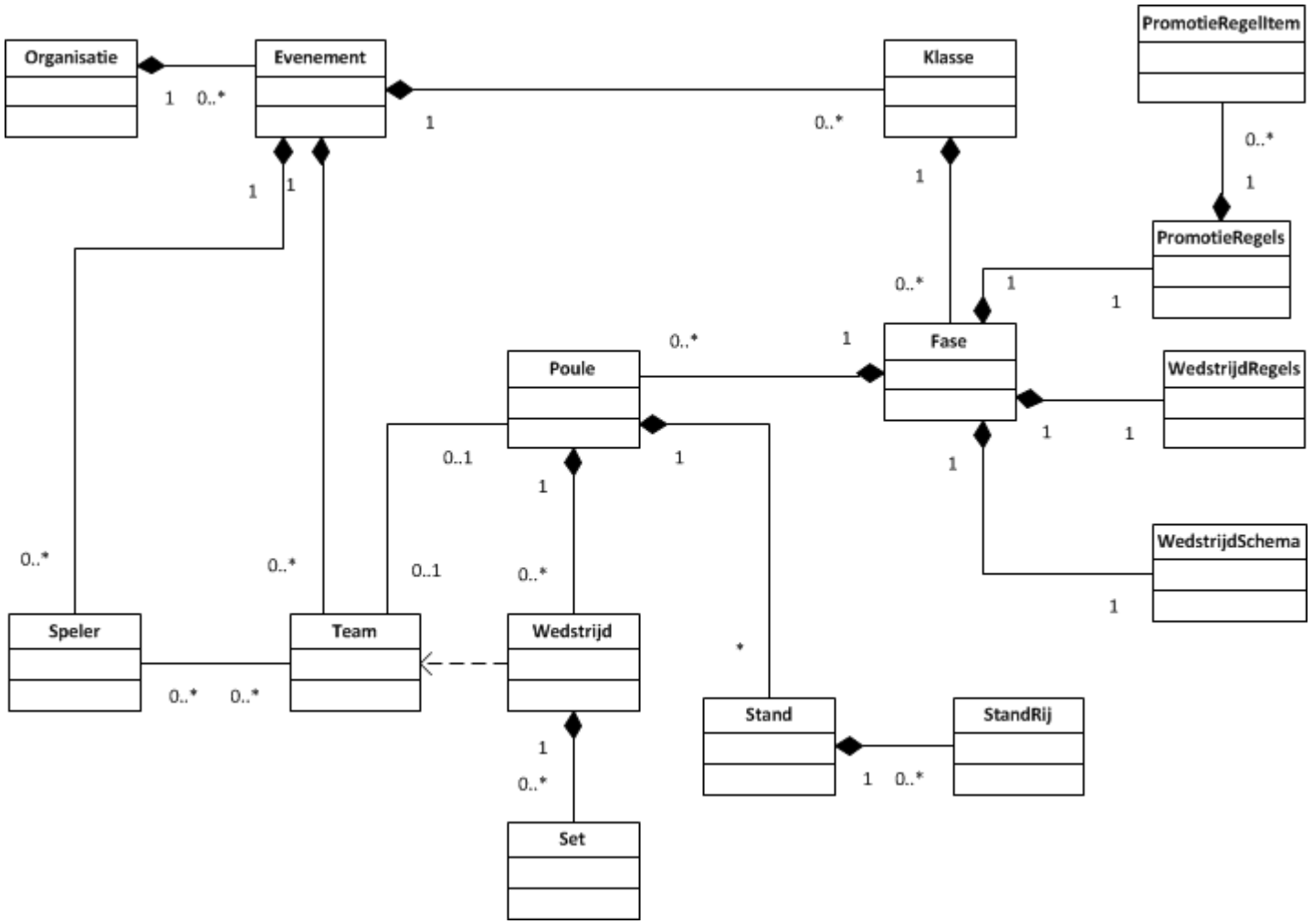
Bijlagen

Bijlage 1: De website beachcompetitie.nl

In 2006 heeft de Nevobo de website http://beachcompetitie.nl gelanceerd. Deze biedt toernooiorganisatoren (inclusief de Nevobo) de mogelijkheid om een toernooi te starten. Vervolgens kunnen spelers/teams zich via de website voor dat toernooi inschrijven.

Als organisator is het mogelijk om de niveaus van dat toernooi aan te geven, wanneer het toernooi plaatsvindt, de inschrijftermijn en dergelijke. Met deze informatie wordt ook meteen een toernooikalender gegenereerd.
De speler kan zich dan voor het gewenste niveau inschrijven en kan door middel van iDeal onmiddellijk betalen. Na de inschrijftermijn krijgt de organisator een helder beeld van het aantal deelnemers per niveau en kan met die gegevens het toernooi organiseren.
Na afloop van het toernooi worden eventueel de standen gepubliceerd en worden de rankings van de spelers in de nationale competitie geupdate.

Bijlage 2: Klassendiagram



Het volledige klassendiagram toont ook een aantal hulpklassen die bepaalde functionaliteit hebben overgenomen om een beter overzicht over het geheel te kunnen houden.