AFSTUDEER SCRIPTIE

Ontwikkelen Ad-2-Add Play-out Monitor

Auteur:
Dhr. M.J.M. van Meurs (99008956)
Haagse Hogeschool

Opdrachtgever:
Dhr. F. Ouwendijk
Media Choice

Bedrijfsmentor:
Dhr. F. Voskamp
Media Choice

Examinatoren:
Dhr. A. van der Molen
Dhr A.G.J. Vinkesteijn
Haagse Hogeschool

Datum:
8 oktober 2004

REFERAAT

M.J.M. van Meurs, 99008956, Afstudeerscriptie, Ontwikkelen Ad-2-Add Play-out Monitor, Naaldwijk, Media Choice, oktober 2004.

Persoonlijk verslag van de ontwikkeling van het Ad-2-Add Play-out Monitor pakket in opdracht van Media Choice te Naaldwijk, uitgevoerd in het kader van het afstuderen aan de afdeling Informatica (opleiding: IVIT, differentiatie: OSTI) aan de Haagse Hogeschool te Den Haag.

Trefwoorden
- MPEG-decoder
- Delphi
- Monitoring
- Ad-Insertion
- Televisie reclame
- RUP
- UML

VOORWOORD

Sinds de aanvang van mijn afstuderen zijn er inmiddels twintig weken verstreken, en heb ik inmiddels de, nu voor u liggende, afstudeerscriptie, welke ter afronding van het afstuderen samengesteld dient te worden, geschreven. Terug kijkend op deze periode kan ik mij niet voorstellen dat de afstudeerperiode alweer bijna op zijn einde loopt. Ik heb een hele prettige tijd gehad bij Media Choice. Het was heel plezierig samen werken met de mensen van dit bedrijf en met alle medewerkers van The Ad-Insertion Platform, waar Media Choice onderdeel van is, in het algemeen.

Via deze weg wil ik graag een aantal mensen bedanken voor hun adviezen, steun en hulp gedurende mijn stage. Allereerst is dat dhr. Ouwendijk, mijn opdrachtgever. Hij was degene die regelmatig met mij mijn voortgang doornam en mij voorzag van tips en ideeën. Voor de technische ondersteuning, op zowel het gebied van het programmeren als van de besturing van de Adtec units, wil ik graag mijn bedrijfsmentor dhr. Voskamp bedanken. De laatste mensen die ik wil bedanken zijn dhr. Van De Molen en dhr. Vinkesteijn. Zij zijn de examinatoren van de Haagse Hogeschool die mij uiteindelijk zullen beoordelen, maar die mij ook hebben geholpen voor aanvang van het afstuderen met suggesties en tijdens het schrijven van de scriptie met feedback hierop.

Den Hoorn, 7 oktober 2004

Mark van Meurs

INHOUDSOPGAVE

1 INLEIDING

De afstudeerperiode loopt inmiddels ten einde en ter afronding van deze periode dient een scriptie geschreven te worden. Deze scriptie omvat een beschrijving van de afstudeeropdracht, samen met een vooruitblik alvorens er aan de opdracht begonnen is. Daarnaast is er een beschrijving terug te vinden betreffende de werkzaamheden gedurende het afstuderen, en een evaluatie hiervan. Tenslotte zijn de leereffecten opgenomen die gedurende het afstuderen zijn opgetreden.

Naar aanleiding van deze scriptie en, in mindere mate, de door mij te houden presentatie en verdediging zullen de examinatoren van de Haagse Hogeschool een beoordeling geven waaraan afgemeten kan worden of er tijdens het afstuderen daadwerkelijk door mij is bewezen te kunnen werken en denken op HBO niveau.

In hoofdstuk 2 van dit stageverslag is een omschrijving gegeven van het stagebedrijf en de opdracht. Er is ingegaan op de structuur van de afdeling, op welke markt de organisatie actief is, met welke nieuwe ontwikkelingen de organisatie zich bezig houdt en de algemene procesgang en procedures van de organisatie. Tevens is er ingegaan op, met betrekking tot de uit te voeren opdracht, wat de probleemstelling is, wat het uiteindelijke doel van de opdracht is, wat ervoor benodigd is om dit te bereiken en welke producten opgeleverd worden.

In hoofdstuk 3 is tot in de details ingegaan op de voorbereidingen die getroffen moeten worden, voordat er aan de opdracht begonnen kan worden. De onderwerpen die met betrekking hierop ter sprake zijn gekomen zijn het kennismaken met de MPEG-2 decoders en het leren gebruiken hiervan en het inrichten van het werkstation.

Nadat de voorbereidingen achter de rug zijn, kan er begonnen worden aan de opdracht. Hoe ik de opdracht gepland heb aan te pakken is beschreven in hoofdstuk 4.

Hoofdstuk 5, 6 en 7 geven een indruk van de door mij gemaakte uitwerkingen van de verschillende ontwerp documenten voorafgaand aan realisatie van de gewenste applicatie. Deze zijn op te delen in het Business Architecture Document, het Requirements Document en het Software Architecture Document.

In hoofdstuk 8 wordt beschreven hoe ik een testplan heb opgesteld om de kwaliteit van de uiteindelijk applicatie te waarborgen.

In hoofdstuk 9 zijn de activiteiten beschreven betreffende het realiseren van de applicatie. Deze realisatie is weer onder te verdelen in de ervaring met de ontwikkelomgeving zelf, het maken van de database, de communicatie met de MPEG-decoders en de interactie tussen de gebruiker en de applicatie.

Wanneer de applicatie uiteindelijk ontwikkeld is, dient deze opgeleverd te worden. Dit aspect is beschreven in hoofdstuk 10.

Hoofdstuk 11 bevat de evaluatie van het proces en het product en tenslotte zijn de leereffecten van de afstudeerperiode terug te vinden in hoofdstuk 12.

2 BESCHRIJVING VAN ORGANISATIE EN OPDRACHT

Dit hoofdstuk bevat een beknopte beschrijving van de organisatiestructuur en de uitgevoerde opdracht. Omdat deze beschrijving voor aanvang van het afstuderen is geschreven is het geschreven in de vorm van een vooruitblik.

2.1 Omschrijving van de organisatie

Deze paragraaf bevat een omschrijving van de organisatie in zijn geheel. Elementen die besproken zijn, zijn een globale omschrijving van The Ad-Insertion Platform (de, relatief kleine, holding waarbinnen Media Choice valt), de afdelingen die binnen de organisatie te onderscheiden zijn, op welke markt zij actief is en de bedrijfsstandaards van algemene procesgang en procedures.

2.1.1 Globale omschrijving

The Ad-Insertion Platform is een holding waarin twee bedrijven nauw samenwerken op het gebied van tv-reclame zendtijd verkoop, apparatuur die benodigd is voor de uitzending van tv-reclame, de productie van tv-spots en de distributie van algemene MPEG-2 content. Het bedrijf is opgericht in 1997.

The Ad-Insertion Platform telt 14 fulltime medewerkers en 1 stagiaire. Doordat het bedrijf relatief weinig werknemers heeft bestaat er geen strakke functiescheidingen.

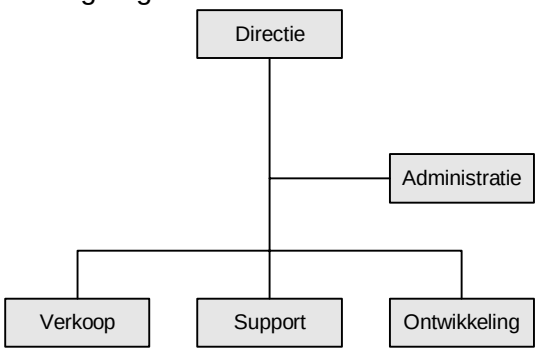
The Ad-Insertion is een, getypeerd aan de hand van de typologie van Mintzberg, ondernemersorganisatie. Het bedrijf wordt bestuurd door de oprichter en enthousiaste directeur en bestaat uit een aantal zeer gemotiveerde medewerkers. Beslissingen worden snel genomen en hierin heeft de directeur altijd het laatste woord. De directeur is ook altijd zeer betrokken bij operationele zaken. Het bedrijf is bijzonder klantgericht.

2.1.2 Structuur van 'The Ad-Insertion Platform'

The Ad-Insertion Platform is op te delen in een vijftal bedrijven:

Bedrijf	**Primaire taak**
Media Choice	Verkoop van zendtijd op (internationale) televisie zenders.
Adtec Northern Europe	Distributie en verkoop van Adtec apparatuur op de Noord-Europese markt.
Video Creative	Productie van televisie reclames
Spotservice en Clipservice	Distributie van televisie reclames en verkoop van video content (videoclips, cartoons, etc.)

Het organigram van The Ad-Insertion Platform ziet er als volgt uit:



Figuur 2-1 Organigram

Er is in deze kleine organisatie toch een duidelijke functiescheiding. Als we echter de functies bekijken die een werknemer uitvoert, komt het vaak uit dat een werknemer zich kan plaatsen in meerdere afdelingen en bedrijven. Het gevolg daarvan is dat niet exact bepaald kan worden hoeveel werknemers op elke afdeling onderverdeeld zijn. Zeker is echter dat mijn plaats in de afdeling onder de afdeling ontwikkeling valt.

2.1.3 Activiteiten van de organisatie

Zoals al eerder vermeld is The Ad-Insertion Platform voornamelijk actief in de wereld van televisie reclame. Sinds 1999 toen een wijziging in de Media Wet het toestond is er begonnen met ‘ad-insertion’. Ad-insertion is het inlassen van advertenties, afkomstig uit het uitzendgebied van de kabelmaatschappij, in de reclameblokken van (inter)nationale televisiezenders.

Adtec Northern Europe is de importeur van de MPEG-2 encoders en decoders van de Amerikaanse fabrikant Adtec Inc. Deze apparatuur maakt het mogelijk om digitale video content af te spelen. Deze organisatie verkoopt MPEG-2 encoders en decoders voornamelijk aan Europese Adtec dealers, maar ook direct aan kabelmaatschappijen, winkels (voor in-store video) en de eigen zuster organisatie Media Choice.

Video Creative is de productie organisatie. Hier worden een groot aantal van de spots die klanten van Media Choice nodig hebben geproduceerd. Het product wat Video Creative verkoopt zijn televisie reclame spots, voornamelijk aan klanten van Media Choice.

Spotservice en Clipservice zijn relatief nieuwe organisaties. Zij zijn verantwoordelijk voor de verkoop van video content (videoclips, extreme sporten films, etc). Tevens dragen deze organisaties zorg voor de distributie van goede doelen spots. Het product wat zij verkopen is video content. Dit gebeurt voornamelijk aan kabelmaatschappijen en aan winkels die in-store video aan willen bieden.

Media Choice is de verkoop organisatie van The Ad-Insertion Platform. De dienst die het verkoopt is zendtijd aan lokaal, regionaal, landelijk en wereldwijd opererende

organisaties. Via Media Choice is het mogelijk te adverteren op internationale televisiezenders in het uitzendgebied van één of meerdere kabel headends. Ook wordt de reclame zendtijd op het CNN Airport Network op Schiphol door Media Choice verkocht.

2.1.4 Algemene procesgang en procedures

De sfeer binnen The Ad-Insertion Platform is goed en informeel. Er is wel een hiërarchische structuur aanwezig, maar door het lage aantal werknemers en de informele sfeer is hier weinig van te merken. Er gelden, net als in andere organisaties, regels en afspraken, maar alles valt te bespreken. Hierdoor is het een zeer flexibele organisatie. De functieverdeling is duidelijk, maar sluit niet uit dat werknemers zich ook met andere zaken bezig houden. Binnen de organisatie geldt de ongeschreven regel dat iedereen een helpende hand uitsteekt waar dit nodig is.

2.2 Uit te voeren opdracht

2.2.1 Bedrijf

Media Choice is een 6 jaar geleden opgericht bedrijf dat door geheel Nederland ‘ad-insertion’ verzorgt. Ad-insertion is het invoegen van lokale/regionale reclameblokken op internationale televisiezenders. Media Choice heeft circa 10 medewerkers.

2.2.2 Probleembeschrijving

Media Choice maakt gebruik van de hardwarematige MPEG-video decoders van Adtec Inc. (ook wel "units" genaamd) om de geprogrammeerde reclame blokken uit te zenden. Deze units zijn door Nederland, België en het Verenigd Koninkrijk verdeelt bij de diverse 'headends' van kabelmaatschappijen. Headends zijn de locaties van kabelmaatschappijen waar vandaan het televisiesignaal verstuurd wordt naar de kabelabonnees. De units zijn op afstand te beheren via Internet of via de telefoonlijn (m.b.v. een modem). Omdat de organisatie momenteel ongeveer 50 units ingezet heeft is het praktisch onmogelijk om met de hand te controleren of alle geprogrammeerde reclame blokken ook daadwerkelijk hebben gespeeld en of de juiste spots zijn uitgezonden. Hiervoor zou er namelijk op iedere unit ingelogd moeten worden om een aantal variabelen en logfiles op te halen om deze te analyseren en te vergelijken.

2.2.3 Doelstelling van de opdracht

Het doel van de afstudeeropdracht is het ontwikkelen van een applicatie die periodiek, meerdere malen per uur, controleert en analyseert of alle remote units de juiste taken uitvoeren en/of uitgevoerd hebben en de resultaten hiervan grafisch presenteert. Dit dient te gebeuren door het ophalen, uit de remote units, van variabelen en logfiles. De opgehaalde data dient daarna verwerkt en opgeslagen te worden in een database. De gegevens in de database worden vergeleken met elkaar (correlatie) en vooraf gedefinieerde waardes. Met de resultaten van deze analyses dienen regelmatig rapporten gegenereerd te worden. Tevens dient er, wanneer er een storing wordt geconstateerd (één of meerder units hebben niet of onvolledig de

vooraf gedefinieerde spots gespeeld), een technische medewerker gealarmeerd te worden.

2.2.4 Nadrukken

De software moet flexibel worden zodat deze, ook bij een wijzigende API van de unit, kan blijven functioneren. Tevens moet de uit te lezen en te analyseren gegevens aanpasbaar zijn, zodat verkoop aan andere gebruikers van Adtec apparatuur mogelijk is. Daarnaast moet het systeem uitbreidbaar blijven om in de toekomst mogelijk uitgebreid te worden zodat relatie interventie mogelijk is in de play-out van de units.

2.2.5 Uitgangssituatie

Voor het uitvoeren van opdracht zijn de volgende middelen benodigd: een relatief moderne PC (min. Pentium 4 met 256Mb geheugen) met als besturingssysteem Windows 2000 of Windows XP. De software die benodigd is zijn: Borland Delphi om mee te programmeren, Rational Rose om UML diagrammen mee te generen en Microsoft Office om rapporten en andere documenten mee te produceren. Ook dient een MPEG-speler van Adtec en een televisie beschikbaar te zijn. Een printer om de diverse rapporten en diagrammen mee uit te printen is ook een vereiste. Documenten en boeken die gebruikt moeten kunnen worden zijn: technische informatie over de Adtec MPEG-speler en één of meerdere boeken over Borland Delphi. Uiteraard moet er ook de beschikking zijn over alle ideeën welke er zijn voor de opdracht.

2.2.6 Concrete werkzaamheden

Om de afstudeerstage tot een goed einde te brengen dienen er een aantal fasen doorlopen te worden:

- Opstellen van een plan van aanpak.
- Het interviewen van diverse medewerkers van Media Choice.
- Verkennen van de mogelijkheden met betrekking tot het uitlezen van de units.
- Maken van een Business Architecture Document.
- Maken van een Requirements Document.
- Maken van een Software Architecture Document.
- Maken van een Test Plan Document.
- Programmeren.
- Testen.
- Schrijven van een beknopte handleiding.

Als methode zal er gebruik gemaakt worden van RUP (Rational Unified Proces). Voor het technisch ontwerpen zal er gebruik gemaakt worden van de UML (Unified Modelling Language) methode. Interview technieken zullen ook gebruikt worden.

Omdat RUP bij aanvang van het afstuderen een nog onbekende methode is bij de afstudeerder is de lijst van de te maken documenten gebaseerd op de lijst van

gemaakte documenten bij een ander, met behulp van RUP, uitgevoerd project waar de afstudeerder de hand op heeft weten te leggen.

2.2.7 Op te leveren producten

Gedurende de loop van het project zullen de volgende producten op geleverd worden:

- Business Architecture Document;
- Requirements Document;
- Software Architecture Document;
- Test Plan Document;
- “Ad-2-Add Play-out Monitor”-applicatie;
- Beknopte handleiding.

3 VOORBEREIDING OP DE OPDRACHT

3.1 Inrichten van de werkplek

Bij aanvang van het afstuderen werd mij een ruime werkplek beschikbaar gesteld. Er stond een computer, unit en televisie. Alles was prima verzorgd, van inrichten was voor mij geen sprake.

3.2 Prepareren van werkstation

3.2.1 Hardwarematige configuratie

De computer die aan mij ter beschikking gesteld is was helaas nog niet volledig bruikbaar. Zo ontbrak de hardeschijf en was er ook geen CD/DVD speler ingebouwd. Aangezien ik geen problemen heb met hardwarematige werkzaamheden en beide componenten op voorraad waren bij Media Choice was dit probleem snel verholpen.

3.2.2 Softwarematige configuratie

Zoals al vermeld zat er in de computer geen hardeschijf toen ik deze kreeg, vandaar dat de computer geheel opnieuw geïnstalleerd moest worden. Ook dit was geen probleem. Alle benodigde software was beschikbaar, dus na een dag installeren had ik de beschikking over een computer met Microsoft Windows XP, Microsoft Office 2003, Borland Delphi 7 en Rational Rose.

3.3 Kennismaking met de MPEG-2 decoders

3.3.1 Aansluiten en verkennen van de apparatuur

Hoewel ik van oorsprong niet veel ervaring heb met videoapparatuur was het aansluiten niet zo een heel groot probleem. Met behulp van een scart-kabel met aan de uitgangen twee audio- en één video ingang was de unit van het type ‘Duet’ snel aan de televisie gekoppeld. Met het meegeleverde verloop stekkertje was het apparaat ook vlot aan de seriële poort van mijn computer gekoppeld en na de installatie van de meegeleverde basis software kon ik aan de gang gaan. Echter, veel meer dan wat basis functies bezat de basis software niet voor het met de hand aan sturen van de unit.

3.3.2 Uploaden van content en verkennen van terminal modus

Om enigszins met de unit te kunnen werken is het vereist dat er een redelijke hoeveelheid content op staat. Omdat het niet mogelijk is om via de seriële poort bestanden naar de speler te uploaden moest er een ethernet verbinding gemaakt worden. Met een ethernet verbinding kan men met het TCP/IP FTP protocol bestanden up- en downloaden naar- en van de unit.

Toen ik via een FTP client de unit gevuld had met data kon ik aan de gang via een zogenaamde terminal modus. Via een telnet sessie kon ik een verbinding leggen tussen mijn computer en de unit. Via deze verbinding kunnen allerlei commando’s gestuurd worden naar de unit. Door het op deze manier experimenteren heb ik redelijk wat kennis opgedaan van de unit API (Application Programming Interface).

3.3.3 Bestuderen van de documentatie

Hoewel via de terminal modus een help functie op te vragen was kreeg ik al snel de behoefte aan wat meer documentatie, aangezien de help file een klein ASCII bestandje was met wat simpele uitleg over de syntax van de commando's en de documentatie map uitgebreide informatie bevatte per commando. De handleiding was een map van ruim 300 A4'tjes waarvan het grootste deel bijlage was met alle mogelijke terminal modus commando's.

3.3.4 Training in het gebruik van de schedules

Schedules zijn de lijsten die de unit vertellen welke spots op welke tijden uitgezonden moeten worden. Hoewel deze omschrijving vrij simpel lijkt zijn ze in de praktijk nog niet zo heel eenvoudig. Met name het feit dat bij ad-insertion niet altijd een exact tijdstip bekend is waarop een spot uitgezonden moet worden maken de schedules complex. Ze hebben namelijk een bepaalde flexibiliteit.

Ad-insertion schedules worden opgebouwd uit blokken. Ieder blok bestaat uit een lijst van spots die uitgezonden moeten worden. Ieder blok krijgt een verwacht tijdstip waarop uitgezonden moet worden en een levensduur waarin het blok geldig is.

Een schedulefile is een, met spatie gesepareerd, ASCII tekst document. Iedere regel in een schedulefile beschrijft één uit te zenden spot. Per uit te zenden spot (per regel) worden de volgende gegevens opgeslagen:

Veld	**Omschrijving**
ScheduleEntry	Code van drie karakters (LOI) om aan te geven dat de regel een schedule item bevat.
UnitID	Vier karakters tellende numerieke representatie van de unit waarvoor het item bedoeld is.
ScheduledTime	Het verwachte tijdstip van uitzending, beschreven als: HHMMSS.
StartWindow	Het tijdstip vanaf wanneer het item uitgezonden zou mogen worden, beschreven als: HHMM.
WindowLength	De tijd die een item heeft na StartWindow om uitegezonden te worden, beschreven als: HHMM.
BreakID	Drie karakters tellende numerieke representatie van het reclame blok waarin het item gespeeld dient te worden.
PositionID	Drie karakters tellende numerieke representatie van het volgnummer welke het item heeft, dit bepaalt de volgorde per reclame blok.
SpotLength	De lengte van een spot, beschreven als: HHMMSS.
ActualPlayedTime	Tijdstip waarop het item uiteindelijk is uitgezonden, beschreven als: HHMMSS
ActualPlayedLength	De lengte van een spot zoals die uiteindelijk is uitgezonden, beschreven als: HHMMSSFF. FF staat in dit verband voor video frames.
ActualPlayedPositionID	Drie karakters tellende numerieke representatie van het volgnummer welke het item uiteindelijk heeft gekregen.
SpotID	Elf karakters tellende naam van de af te spelen spot, waarbij de eerste drie karakters altijd nul zijn (000).

ValidationCode	Vier karakters tellende code die beschrijft of het afspelen van een spot goed is gegaan of welke problemen opgetreden zijn.
Comments	Oneindig lang veld om commentaar aan een schedule item toe te voegen.

De velden die beginnen met ‘Actual’ en ‘ValidationCode’ zijn velden die nooit ingevuld zijn in een schedulefile. De units, echter, bieden de mogelijkheid om, nadat een volledig schedulefile is doorlopen, deze te kopiëren naar een, zogenaamde, ready file. In deze ready file worden deze velden wel ingevuld. Bij aanvang van het afstuderen worden de ready files gebruikt om de werking van de units te controleren. Echter, dit wordt als te omslachtig ervaren en dit zorgde er soms voor dat, bij gebrek aan tijd om dit dagelijks te controleren, de signalering van fouten soms dagen na optreden plaats vond.

Bovenstaande informatie over schedules was niet als zodanig uit de documentatie van de units te halen. Vandaar dat ik mijn bedrijfsmentor heb gevraagd mij hierover een korte training te geven, aangezien dit wel cruciale informatie was om het afspelen van spots te controleren.

3.3.5 Training in het lezen van de logfiles.

Hoewel minder complex als de schedules had ik ook enige uitleg nodig bij de logfiles. De logfiles van de units bestaan namelijk voor een deel uit codes die niet zomaar te lezen zijn. Echter, na een uurtje uitleg hierover door mijn bedrijfsmentor was ook dit begrijpelijk voor mij.

Logfiles zijn net als schedulefiles ASCII tekst bestanden, echter in dit geval gescheiden door komma’s. Hoewel er vrij veel activiteit op de unit gelogd wordt, is het voor mij uitsluitend van belang me te richten op activiteiten die daadwerkelijk het afspelen van de units beschrijven. Deze log regels zien er als volgt uit:

Veld	**Omschrijving**
LogCode	Zes karakters tellende numerieke waarde om aan te geven wat voor activiteit gelogd is.
Time	Het tijdstip waarop de logactiviteit plaats vond, beschreven als: HH:MM:SS.
StartDiscID	Twee karakters tellende numerieke representatie van de fysieke hardeschijf/CD-Rom/DVD-Rom waarop de spot, waar het logitem mee begon, zich bevond.
StartPartitionID	Twee karakters tellende numerieke representatie van de partitie op de betreffende disk waarop de spot, waar het logitem mee begon, zich bevond.
StartFileID	Drie karakters tellende numerieke representatie van de spot, waar het logitem mee begon, zich bevond op de betreffende disk en partitie.
StartSpotName	Acht karakters tellende naam van de spot waar het logitem mee begon.
StartExtension	Drie karakters tellende extensie van de spot waar het logitem mee

	begon.
EndDiscID	Twee karakters tellende numerieke representatie van de fysieke hardeschijf/CD-Rom/DVD-Rom waarop de spot, waar het logitem mee eindigde, zich bevond.
EndPartitionID	Twee karakters tellende numerieke representatie van de partitie op de betreffende disk waarop de spot, waar het logitem mee eindigde, zich bevond.
EndFileID	Drie karakters tellende numerieke representatie van de spot, waar het logitem mee eindigde, zich bevond op de betreffende disk en partitie.
EndSpotName	Acht karakters tellende naam van de spot waar het logitem mee eindigde.
EndExtension	Drie karakters tellende extensie van de spot waar logitem mee eindigde.
Description	Onbeperkt aantal karakters tellend veld wat de activiteit in ‘gewone’ taal omschrijft.

De logfile beschrijft steeds twee spots, omdat de unit begint met het opstellen van een logentry wanneer de activiteit start. Als gedurende de activiteit de unit een andere spot laadt zal de activiteit afgesloten worden met een andere spot, die dan ook gelogd wordt.

4 UITWERKEN VAN HET PLAN VAN AANPAK

4.1 Keuze van de opdracht

De opdracht is door Media Choice aangeboden. Het bedrijf was opzoek naar een applicatie waarmee het in staat zou kunnen zijn om 24/7 in de gaten te kunnen houden wat haar units in de diverse headends van de kabelmaatschappijen aan het doen zijn en hebben gedaan. Ik heb deze opdracht aangenomen, omdat het voor mij een aantal interessante elementen bevatte. Naast het ‘gewone’ administratie gedeelte (opslaan, wijzigen en verwijderen van gegevens in een database), wat het moest gaan omvatten, had het ook enige technische aspecten die mij zeer interessant leken. In het bijzonder de communicatie met de units sprak mij aan, omdat ik hiervoor een nieuw protocol zou moeten gaan implementeren.

4.2 Verzamelen van informatie

De informatie, aangeboden tijdens de aanbieding van de opdracht zelf, was de eerste bron. Hiervan heb ik veel kunnen gebruiken. Ten tweede heb ik een gesprek gehad met mijn opdrachtgever en mijn bedrijfsmentor. Deze informatie complementeerde het voor mij. Tevens vertelde mijn opdrachtgever mij over het ontstaan van The Ad-Insertion Platform, de activiteiten en de toekomst plannen. Hiermee heb ik het kader van de opdracht, de probleemstelling, de doelstelling en de op te leveren productenlijst kunnen samenstellen.

Gezien het feit dat ik geen ervaring had met RUP ben ik tijdens het opstellen van de opdrachtomschrijving al op zoek gegaan naar documenten over dit onderwerp. Tijdens deze zoektocht heb ik de hand weten te leggen op de documenten behorende bij een ander project wat met behulp van RUP uitgevoerd was. Aan de hand van deze documenten en enige ervaring heb ik een lijst op kunnen stellen van de uit te voeren werkzaamheden en de planning.

4.3 Keuze voor methoden en technieken

Gezien het feit dat Media Choice al in het bezit was van Borland Delphi 7 in combinatie met Rational Rose als ontwikkelomgeving, was voor mij snel de keuze gemaakt om object georiënteerd te werk te gaan. Delphi is namelijk een object georiënteerde taal. Rose is een pakket waarmee het mogelijk is om met behulp van UML (Unified Modelling Language) een object georiënteerd systeem te ontwerpen. Mijn eerste plan was om SDM te gebruiken als project beheersmethode. Echter, na een gesprek met een collega ICT’er heb ik hiervan afgezien. SDM (System Development Method) gaat uit van de waterval methode, iets wat in principe niet strookt met het feit dat de software in de toekomst mogelijk nog uitgebreid gaat worden. Hij raadde mij aan gebruik te maken van RUP (Rational Unified Proces). Dit is een project beheersmethode die uitgaat van een iteratieve ontwikkeling en daarnaast sterk geïntrigeerd is met UML. Dit maakt een incrementele ontwikkeling mogelijk.

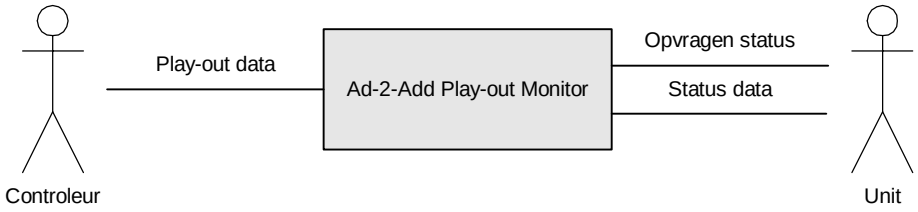
RUP werkt met zogenaamde artefacten. Artefacten zijn stukjes informatie of software die geproduceerd worden in een software ontwikkel traject. Een model kan een artefact zijn, maar ook een beschrijving, een applicatie of een volledig document. Daarnaast kan een document ook meerdere artefacten bevatten. RUP kent een tiental verschillende artefacten. Het is aan de software ontwikkelaar te bepalen welke artefacten gemaakt worden in het ontwikkel proces. Hierbij is de mogelijke meerwaarde van een artefact een doorslaggevende factor. Aangezien RUP, bij aanvang van het afstuderen, nieuw was, ben ik uitgegaan van de artefacten die gemaakt werden bij een min of meer gelijkwaardig project waar ik tijdens mijn onderzoek naar RUP de hand op heb weten te leggen.

5 MAKEN VAN HET BUSINESS ARCHITECTURE DOCUMENT

Om zicht te krijgen op de bedrijfsomgeving waarbinnen het te ontwikkelen systeem gaat functioneren heb ik een Business Architecture document gemaakt. Dit model beschrijft wat het probleem is wat opgelost dient te worden, het beschrijft niet hoe het opgelost moet gaan worden. Het Business Architecture document behoort niet tot de technische documentatie van de Ad-2-Add Play-out Monitor (afgekort: APM), wel stelt het de achtergrond en de context voor de applicatie vast. Het Business Architecture document dient als het model van de organisatie waar APM ingezet zal worden.

5.1 Opstellen van het domein model context diagram

Ik ben begonnen met het maken van een context diagram van het systeem. Aan de hand van de eisen en wensen van de opdrachtgever heb ik een analyse kunnen maken van de entiteiten buiten het systeem en de datastromen die tussen het systeem en de entiteiten liggen. Omdat het het eerste diagram is wat het systeem beschrijft was het niet een eis dat alle mogelijke verschillende entiteiten en datastromen voorkomen in de figuur. Het is in eerste instantie een diagram om een versimpeld zicht op het systeem en de context te geven.



Figuur 5-1 APM context diagram

5.2 Definiëren van de use cases

Na het opstellen van het domein model context diagram ben ik begonnen met, aan de hand van dit context diagram, het definiëren van de use cases. Binnen UML worden use case diagrammen gebruikt om de functionele eisen aan het systeem weer te geven. In het vervolg van de systeemontwikkeling worden de informele use cases gebruikt bij het opstellen van de formelere sequence diagrammen.

Aan de hand van de opgestelde context diagram kon ik de volgende use cases onderscheiden:

Use Case ID	Titel
0001	Opvragen unit status
0002	Invoeren unit status data
0003	Verstrekken playout data

Hoewel het aantal use cases voor APM sowieso niet groot is heb ik het aantal use cases in het Business Architecture document bewust tot een minimum beperkt. Hier golden de use cases uitsluitend als onderdeel van een globale beschrijving van het probleem.

In kort houden deze use cases het volgende in:

5.2.1 Use Case 0001 – Opvragen unit status

Indien er minimaal één unit in het systeem geconfigureerd dan vraagt het systeem aan alle geconfigureerde units hun nieuwste log gegevens.

5.2.2 Use Case 0002 – Invoeren unit status data

Indien een unit van het systeem een verzoek heeft ontvangen om log gegevens te verstrekken zal deze de gevraagde gegevens invoeren. Het systeem slaat deze vervolgens op in een database.

5.2.3 Use Case 0003 – Verstrekken play-out data

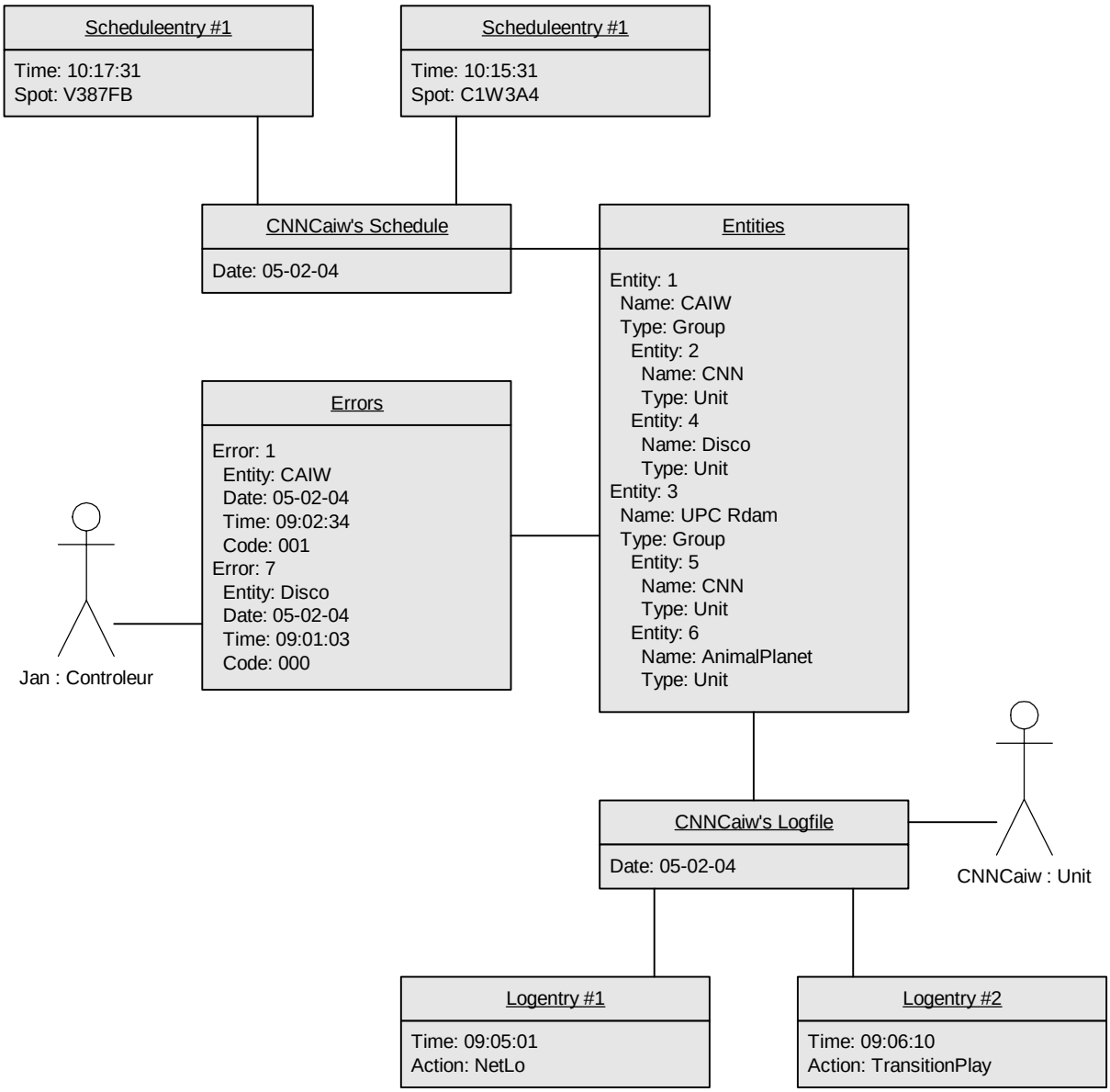
Indien het systeem van minimaal één unit in het systeem log gegevens ontvangen heeft zullen deze door het systeem geanalyseerd worden. De resultaten van deze analyse inclusief de brongegevens zullen aan de controleur gepresenteerd worden.

5.3 Opstellen van een voorbeeld domein object model

Aan de hand van het context diagram en de use cases ben ik een voorbeeld object model gaan maken. Dit object model is niet gebaseerd op werkelijke gegevens, het is uitsluitend bedoeld om een beeld te krijgen welke data in het systeem opgeslagen wordt en welke relatie deze data met elkaar heeft.

Dit object model heb ik gemaakt aan de hand van mijn visie op de uitwerking van de opdracht. Al vrij snel had ik, onder andere na gesprekken met mijn bedrijfsmentor, een idee van hoe ik de data zou structureren. Uitgaande van de opmerking die ik nog niet lang geleden hoorde van een docent dat klasse diagrammen in principe één-op-één te vertalen zijn naar ERD’s, ben ik dit om gaan draaien en heb ik een klasse diagram en object model gemodelleerd naar mijn visie op de uiteindelijke database.

Dit is niet de juiste methode geweest (zie ook hoofdstuk 11). De juiste methode was geweest als ik, uitgaande van de tekst van de probleemomschrijving en de use cases, zelfstandig naamwoorden was gaan onderscheiden als kandidaat klassen. Eventueel had dit gecomplementeerd kunnen worden met een brainstormsessie.



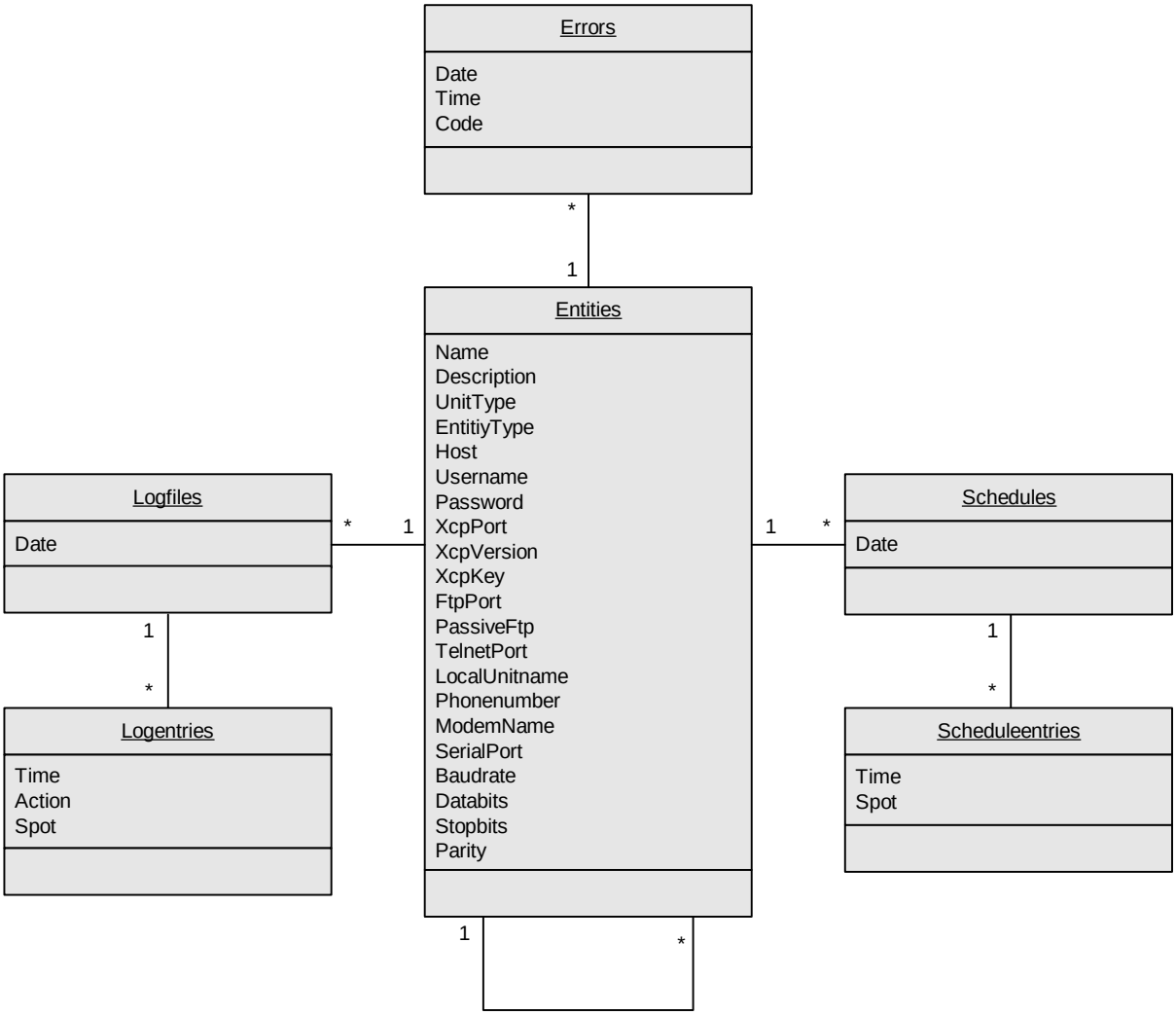
Figuur 5-2 Domein object model voorbeeld

CNNCaiw is een unit die op verzoek van het systeem zijn ‘logfiles’ er in invoert. Deze logfiles bestaan uit diverse ‘logentries’ waarin gebeurtenissen in de unit op een bepaald moment beschreven staan. De logfiles staan gekoppeld aan de ‘entities’. In de entities wordt beschreven welke units aan het systeem gekoppeld zijn en hoe. In dit object worden de in de logfile beschreven logentries vergeleken met de ‘schedules’ en ‘scheduleentries’ en bepaald of deze geen onverwachte informatie bevatten en/of de verbindingen naar de units nog intact zijn. Indien dit niet het geval is wordt in ‘errors’ een nieuwe regel toegevoegd waarin beschreven wordt welke error bij welke entiteit hoort en wat de error inhoudt. Jan is controleur en heeft een continu zicht op de errors en kan ingrijpen wanneer een error optreed.

In het object model heb ik gekozen voor het object ‘entities’. Dit is een niet veel zeggende naam. Ik heb dit object gemaakt, omdat deze meerdere rollen aan kan nemen. Ten eerste kan het een unit omschrijven, maar het kan ook een groep units (een headend) omschrijven. Iedere groep units kan tevens weer andere groepen units omschrijven. Vandaar ook de relatie met zichzelf.

5.4 Afleiden van het klasse diagram

Met het voorbeeld object model in de hand heb ik een klasse diagram gemaakt.



Figuur 5-3 Klasse diagram

Zoals ook in het object model bestaat de klasse ‘entities’ in dit diagram. Echter, een aantekening is hier vereist. De indruk zou gewekt kunnen worden dat ‘logfiles’ en ‘schedules’ aan alle soorten ‘entities’ gekoppeld kan worden. Dit is niet waar. ‘Logfiles’ en ‘schedules’ kunnen alleen een relatie hebben met ‘entities’ van het type unit. ‘Errors’ daarentegen kunnen wel aan alle soorten ‘entities’ gekoppeld worden.

5.5 Bespreking Business Architecture document

Met mijn bedrijfsmentor heb ik het Business Architecture document besproken. Ten eerste had hij zijn twijfels over de klasse ‘entities’. Hij vroeg zich af of het niet verstandiger zou zijn om twee verschillende klassen te introduceren. Één klasse die de units beschrijft en één klasse die de unit groepen beschrijft. Echter, mijn bezwaar daartegen was dat beide klassen dan zeer veel overlappende gegevens zouden bevatten, daar het de bedoeling is dat door middel van overerving units die een relatie hebben met een groep connectie gegevens ophalen uit de groep beschrijving. Hierbij dacht ik dus vanuit het perspectief van een ERD in plaats vanuit het klasse diagram. Ten tweede had mijn bedrijfsmentor een aantal opmerkingen over duidelijkheid en spelfouten, deze heb ik in een nieuwe versie verbeterd.

6 MAKEN VAN HET REQUIREMENTS DOCUMENT

Nadat ik het Business Architecture document had gemaakt was het tijd voor het Requirements document. In dit document wordt beschreven wat de achtergrond van het project is, hoe de gewenste situatie eruit ziet, hoe de userinterfaces eruit gaan zien en wat de gedetailleerde eisen aan het systeem zijn.

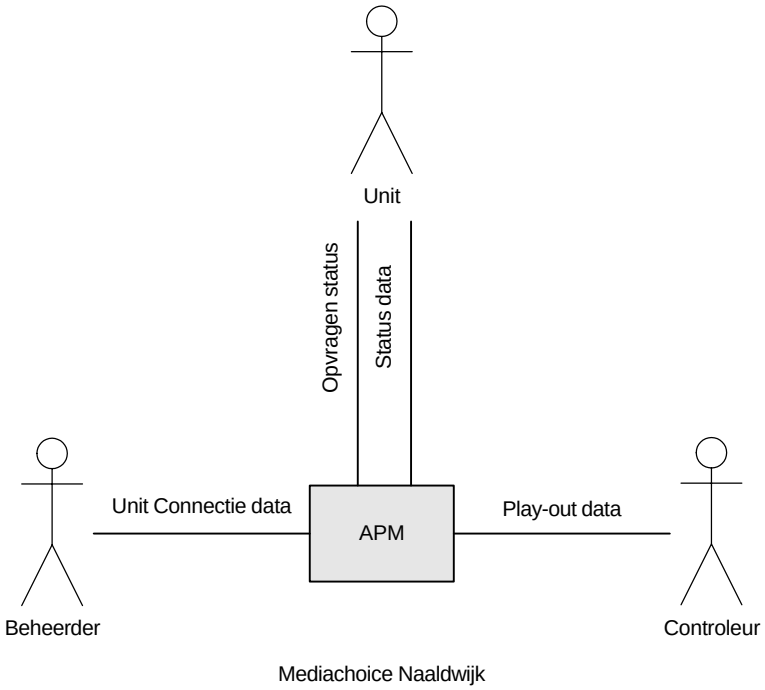
6.1 Opstellen van de project visie

In de project visie wordt de uiteindelijke missie van het project vanuit het perspectief van het bedrijf beschreven. Het opstellen van deze visie was vrij snel gedaan. Uitgaande van de opdrachtomschrijving was het grootste gedeelte van de project visie indirect of direct hieruit af te leiden. Het enige onderdeel van de project visie wat niet direct of indirect uit de opdrachtomschrijving was af te leiden waren de succes criteria voor het bedrijf. Deze criteria zijn de volgende:

- Het uiteindelijke systeem zal inzicht geven in de status van alle ingestelde units;
- De statusdata is maximaal 1 uur geleden voor het laatst bijgewerkte;
- Het systeem is in principe 24/7 beschikbaar;
- De maximale downtime van het systeem is 5 uur per.

6.2 Definiëren van de toekomstige use cases

Het use case model beschrijft het toekomstige systeem doormiddel van het modelleren van de interne bedrijfsprocessen uitgaande van een werkend systeem in de toekomstige situatie. Het beschrijft de interne systemen en de verschillende gebruikersrollen die deel uitmaken van de processen binnen het systeem. Het richt zich voornamelijk op de interactie tussen APM en de actoren (een entiteit die buiten het systeem staat en die direct communiceert met het systeem) binnen het systeem. Allereerst heb ik het context diagram uit het Business Architecture document erbij genomen en verder uitgewerkt met meer details.



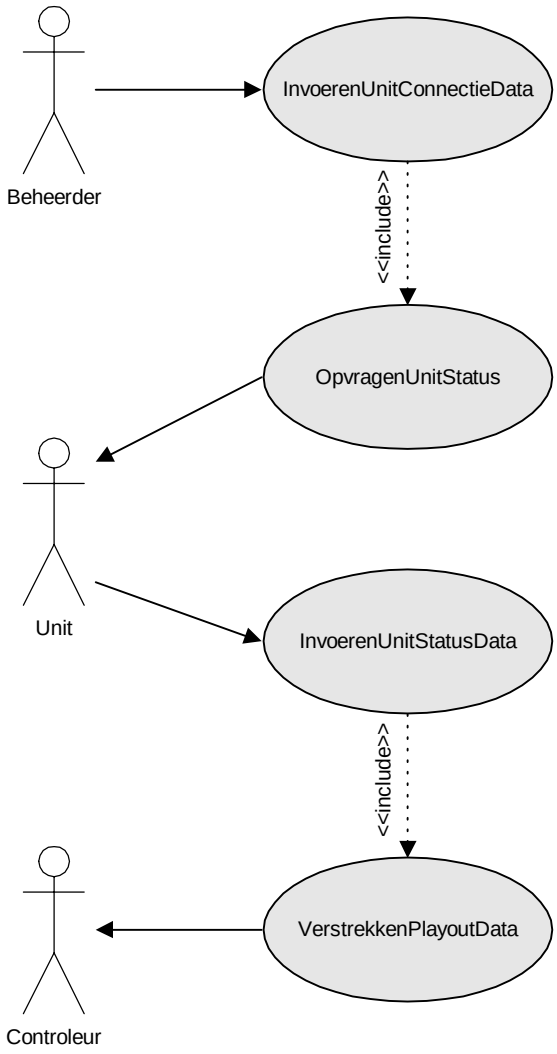
Figuur 6-1 Toekomstige situatie use case context diagram

Zoals te zien is is er een actor bijgekomen, namelijk ‘beheerder’. In eerste instantie had ik de rol van beheerder buiten beschouwing gelaten, omdat dit een ondersteunde actor is, een actor die uitsluitend indirect bijdraagt aan een correcte werking van het systeem. Echter, omdat het Requirements Diagram een hogere mate van detail vereist heb ik deze actor toegevoegd. Met deze actor hangen ook use cases samen. De use cases lijst ziet er dan als volgt uit:

Use Case ID	Titel
0001	Invoeren unit connectie data
0002	Opvragen unit status
0003	Invoeren unit status data
0004	Verstrekken playout data

Ten opzichte van het Business Architecture document is de use case ‘invoeren unit connectie data’ toegevoegd. Deze use case houdt niet meer en niet minder in dan het invoeren van alle gegevens die benodigd zijn om een unit of een groep units vanuit het systeem te kunnen benaderen.

De use cases zien er dan, gemodelleerd in een use case diagram, als volgt uit:



Figuur 6-2 Gewenste situatie use case

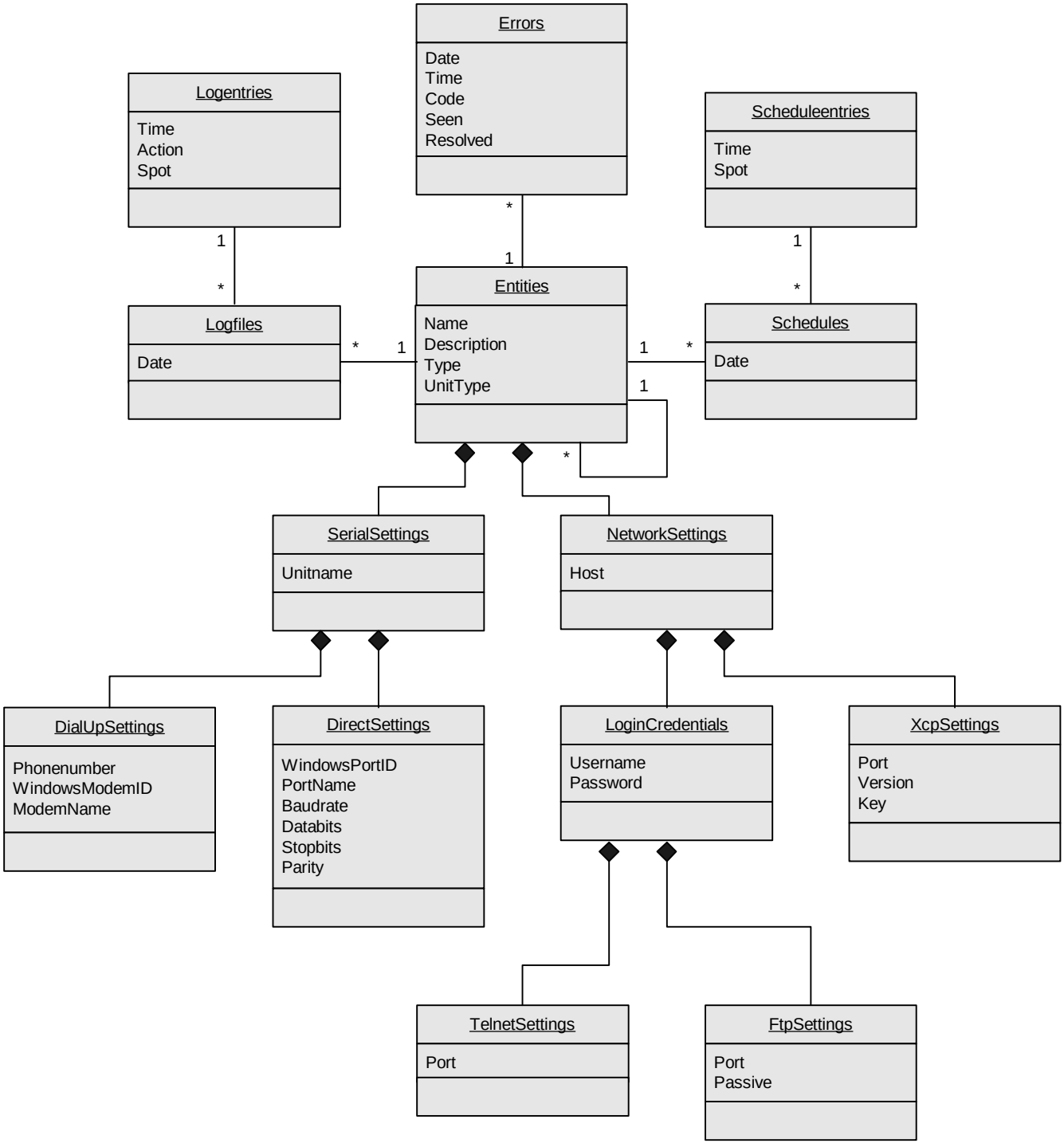
6.3 Opstellen van het use case object model

Na het toevoegen van de actor ‘beheerder’ veranderde ook het object model. De wijziging hierbij is echter zo klein dat het opnieuw opnemen van dit diagram overbodig is. Ik heb een actor ‘Beheerder’ ingevoegd en deze heeft een relatie met ‘entities’.

6.4 Opstellen van het use case klasse diagram

Hoewel het object model niet veel veranderde heb ik in het Requirements diagram het klasse diagram wel onderhanden genomen. In het Business Architecture document had ik nog alle gegevens van ‘entities’ in één klasse samen genomen. In het Requirements document ben ik dit gaan splitsen. Ik heb hiervoor gekozen, omdat in principe een unit of unit groep niet alle informatie hoeft te bevatten. In theorie is het mogelijk dat een unit groep uitsluitend een ‘name’ en ‘type’ als attribuut ingevuld heeft. Een unit zou, in het meest minimale geval, af kunnen met uitsluitend ‘name’,

'type', 'unitname', 'phonenumber', 'windowsmodemid' en 'modemname'. Het nieuwe klasse diagram ziet er als volgt uit:



Figuur 6-3 Use case model klasse diagram

Ik heb gebruik gemaakt van compositie relaties omdat de klassen die in de figuur onder 'entities' liggen allemaal subklassen zijn van deze klasse. Met elkaar vormen ze de representatie van één unit of unit groep.

6.5 Opstellen van de userinterface specificaties

Door de actors en use cases te analyseren was het voor mij mogelijk om de specificaties van de toekomstige userinterface op te stellen.

6.5.1 Globaal gebruiksscenario

In overleg met mijn bedrijfsmentor heb ik een korte samenvatting gemaakt van hoe de userinterface formulieren globaal zullen werken en met elkaar samenhangen.

Afhankelijk van de rol van de gebruiker wordt er bij het starten van de applicatie een keuze gemaakt van wat er gedaan moet worden. Standaard zal de applicatie gestart worden met het controle scherm voor de controleur.

Indien de gebruiker de rol van beheerder heeft kan hij/zij ervoor kiezen uit het pulldown menu de beheermodule te kiezen. Hij/zij zal dan een boomstructuur gepresenteerd krijgen met daarin alle units onderverdeeld in unit groepen. Hier kunnen units en unit groepen aangemaakt, gewijzigd en verwijderd worden.

Indien de gebruiker de rol van controleur heeft hoeft hij/zij na het opstarten niets te doen. De controlemodule toont de gebruiker een boomstructuur met alle units onderverdeeld in unit groepen. Daarnaast wordt er een lijst getoond met alle te verwachten acties. Hierin kan een controleur bijvoorbeeld zien dat er ad-inserts gepland staan met de bijbehorende starttijd. Ook is er een lijst te zien met alle geconstateerde fouten en een kaart van Nederland met daarop de locaties waarop een unit of unitgroep met problemen zich bevindt.

6.6 Opstellen van de gedetailleerde systeem eisen

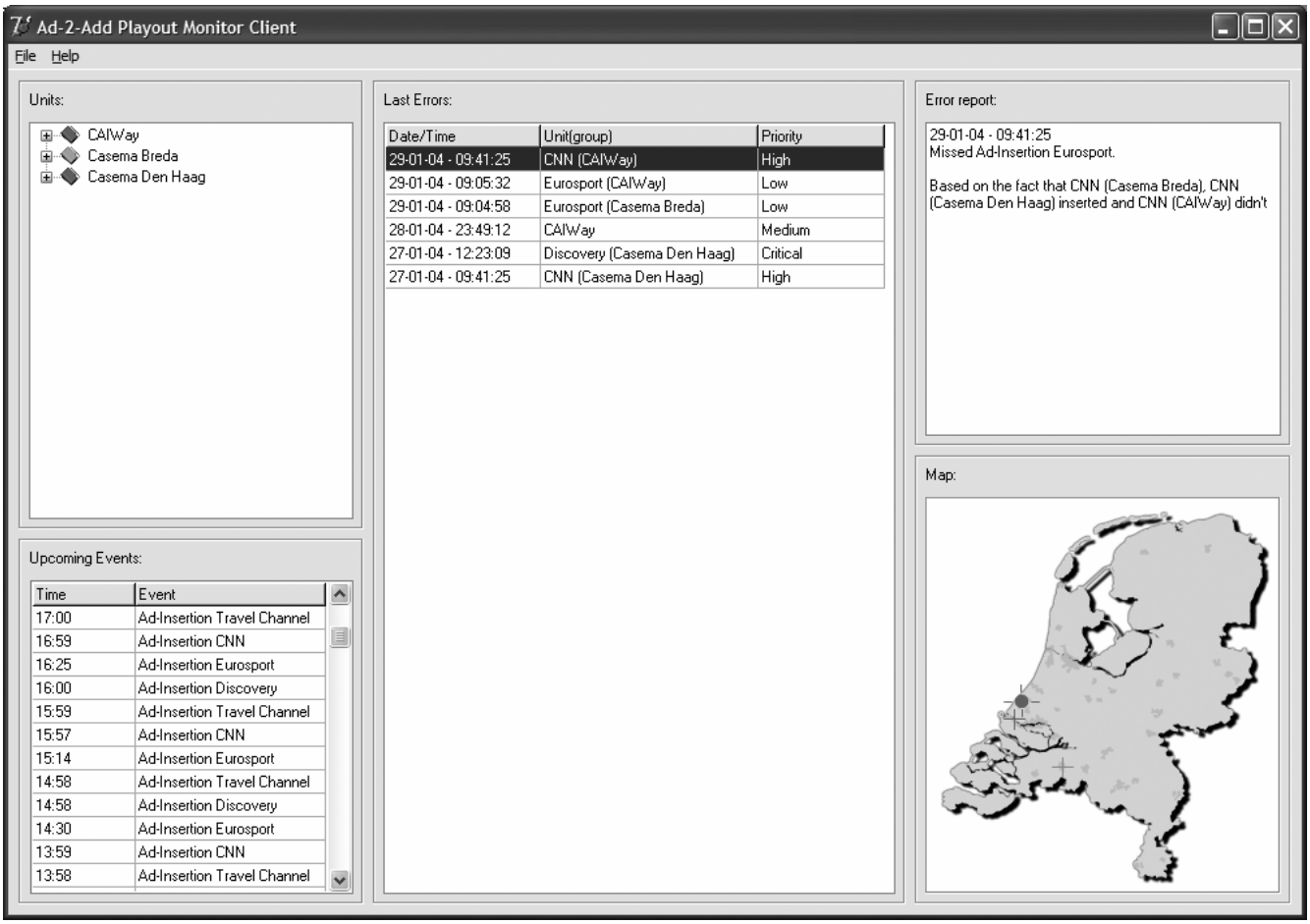
Om een goed zicht te krijgen welke eisen er aan het systeem gesteld worden en om een ‘checklist’ te hebben aan de hand waarvan het systeem gecontroleerd kan worden heb ik de gedetailleerde eisen die aan het systeem gesteld worden op een rijtje gezet. In de volgende tabel maak ik gebruik van een aantal afkortingen:

- **BR - Business Rule eisen**: alle eisen die er zorg voor dragen dat uitsluitend samenhangende gegevens ingevoerd kunnen worden (voorbeeld: wanneer aangegeven wordt dat er geen telefonische verbinding met de unit gemaakt kan worden is het zinloos om een modem te definiëren om verbinding via te maken) en er geen sprake kan zijn van conflicterende data tijdens het invoeren.
- **FR – Functional eisen**: alle eisen die er zorg voor dragen dat de applicatie logisch functioneert met betrekking tot het tonen van de userinterface en de gebruiker dwingt correct te werk te gaan. Voorbeeld: wanneer vereist wordt dat een veld ingevuld wordt voordat door gegaan mag worden zal een ‘Next’ knop uitgeschakeld moeten zijn totdat het betreffende veld door de gebruiker ingevuld is.

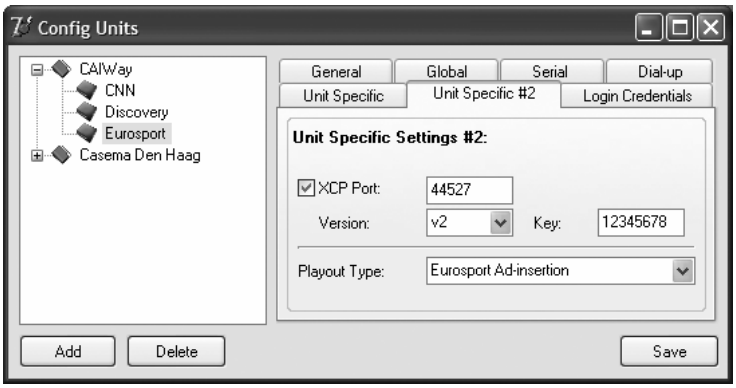
- **PR - Proficiency eisen**: alle eisen met betrekking tot beveiliging, concurrency, maximale service tijd, etc.

RequirementID	Rule
BR-32-01-0001	NetworkSettings en SerialSettings kunnen uitsluitend gedefinieerd worden als er een Entity gecreëerd is.
BR-32-01-0002	DirectSettings en DialUpSettings kunnen alleen gedefinieerd worden als er een SerialSetting voor gecreëerd is.
BR-32-01-0003	LoginCredentials en XcpSettings kunnen alleen gedefinieerd worden als er een NetworkSetting gecreëerd is.
BR-32-01-0004	TelnetSettings en FtpSettings kunnen allen gedefinieerd worden als er een LoginCredential gedefinieerd is.
BR-32-02-0000	Name mag in iedere groep of subgroep slechts één keer voorkomen.
FR-341-0001-000	In de configuratie schermen kunnen alleen connectie gegevens ingevoerd worden als het verloop conform BR-32-01-0001 t/m BR-32-01-0004 is.
PR-413-0001-000	Login gegevens om de units te benaderen moeten gecodeerd opgeslagen worden.
PR-43-0001-001	De APM server module moet 24/7 beschikbaar zijn.
PR-43-0001-002	De down tijd van de server module mag maximaal 5 uur per maand zijn.
PR-44-0001-000	Optimistic concurrency wanneer 2 records gelijktijdig worden geüpdate.
PR-46-0001-000	De poll interval moet instelbaar zijn. Zowel de standaard interval als de ‘hotinterval’ (moment dat een event verwacht wordt op een bepaald kanaal).

Naast de tekstuele eisen heb ik ook in de gedetailleerde eisen prototypes van de toekomstige userinterfaces opgenomen. Deze userinterfaces zijn deels gebaseerd op eerder gemaakte requirements, klasse diagrammen en use cases, maar ook voor een heel groot deel op overleg met de opdrachtgever en de bedrijfsmentor. Samen met hen heb ik de door mij gemaakte schets uitgewerkt tot de hier getoonde prototypes. Opvallend is bijvoorbeeld de kaart van Nederland. In een eerder stadium is hier nog niet over gesproken, maar tijdens dit gesprek met de opdrachtgever bleek dat hij behoefte had aan een grafische weergave van foutmeldingen. Omdat deze kaart de werking van de applicatie niet drastisch veranderd heb ik besloten dit nog in deze versie van APM op te nemen.



Figuur 6-4 Form-0001-0000 – controle module userinterface



Figuur 6-5 Form-0002-0006 – beheer module userinterface

7 MAKEN VAN HET SOFTWARE ARCHITECTURE DOCUMENT

In het Software Architecture document heb ik alle interne beslissingen beschreven die benodigd zijn om het systeem te implementeren. Dit heb ik gedaan aan de hand van een zestal ‘views’. Iedere view zich richt op een ander aspect van het systeem. Het doel is te beschrijven wat de belangrijkste onderdelen van het systeem zijn, hoe het is gestructureerd, wat de systeem proces stromen zijn, en wat de belangrijkste interfaces zijn. Vanuit een hoger niveau is het doel om inzicht te krijgen in het systeem vanuit verschillende perspectieven en daardoor alle kritieke systeem functionaliteiten in kaart te brengen.

De views welke ik heb gehanteerd zijn:

- Deployment view;
- Logical view;
- Data view;
- Process (concurrency) view;
- Implementation view;
- Security view.

7.1 Richtlijnen opstellen

Voordat ik begon met het uitwerken van de diverse views heb ik eerst architectuur richtlijnen opgesteld. Aan de hand van deze richtlijnen zal het systeem opgebouwd gaan worden. De richtlijnen lijken in eerste instantie misschien voor de hand liggend, maar toch vond ik het prettig om alles op papier te zetten zodat deze allemaal een keer geïnventariseerd worden. Deze richtlijnen zijn ook weer onder te verdelen in een aantal categorieën: algemeen, userinterface laag, business laag en data laag.

Enkele voorbeelden zijn:

Algemeen

- Minimaliseer de afhankelijkheid van een bepaalde tool of een bepaalde leverancier.
- Maximaliseer de invoering van beproefde standaarden en technologieën.
- Ontwerp voor het op langere termijn toepassen van principes als:
    - Onderhoudbaar vanaf één plaats
    - Configureerbaar, uitbreidbaar
    - Schaalbaar

Userinterface laag

- De UI zal verborgen en onverborgen edit checks uitvoeren (bijvoorbeeld controle op correcte invoer), controle op maximum lengte en validatie van vereiste attributen. Dit alles wordt gebaseerd op een meta data interface die geëxporteerd wordt vanuit de business laag.

Business laag

- De business laag geeft waarden van business objecten terug. De grootste verantwoordelijkheid van de business laag is om de illusie te geven aan de UI laag dat alle business objecten in het fysieke geheugen staan.
- Dit houdt tevens in dat de business laag interface geen informatie over het DB schema prijsgeeft.

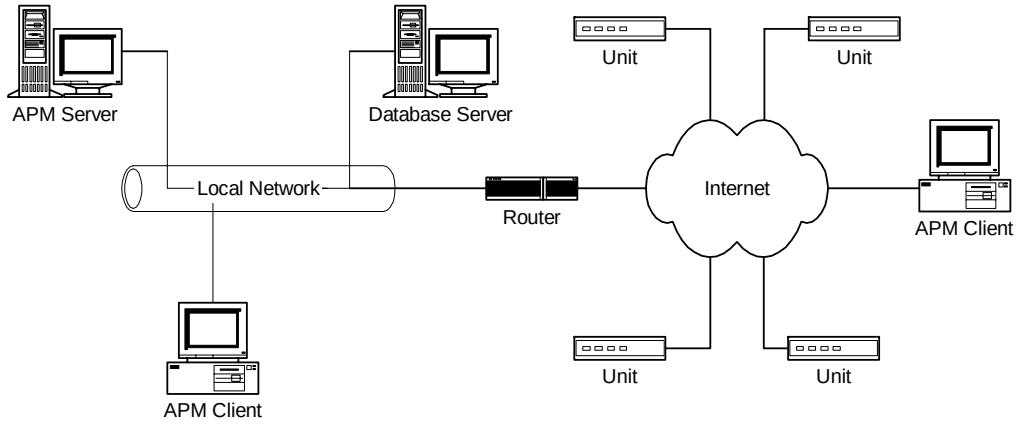
Data laag

- De datalaag omvat volledig het indelen van business attributen naar relationele tabellen. Dit maakt het mogelijk dat de database wordt herontworpen zonder dat business of gebruiker services worden aangepast.

7.2 Uitwerken deployment view

Het deployment view documenteert de fysieke topologie van het systeem zoals gemodelleerd in het deployment model. Er wordt in beschreven hoe computers worden geïmplementeerd en hoe de connectie tussen deze computers is. De configuratie van ieder onderdeel wordt ook beschreven; operating system, database, COTS (commercial off the shell = materiaal beschikbaar op de markt) en programmatuur.

Aan de hand van gesprekken die ik met de opdrachtgever en bedrijfsmentor heb gehad met betrekking tot de uiteindelijke topologie heb ik de volgende figuur gemaakt:



Figuur 7-1 Fysieke en logische verbindingen die samen het APM systeem vormen

Uiteraard is dit een voorbeeld van hoe het eruit zou kunnen zien. In een productie situatie zouden bijvoorbeeld de database server en de APM server op één systeem kunnen draaien en zou er, bijvoorbeeld, sprake kunnen zijn van meerdere clients op het lokale netwerk. Het gaat er in deze figuur in eerste instantie om dat duidelijk is dat de APM server en client twee gescheiden onderdelen moeten worden die met elkaar over een TCP/IP netwerk kunnen communiceren. Daarnaast kan deze figuur

bijdragen in het onderscheiden van de verschillende componenten die met elkaar het systeem vormen.

Aan de hand van de componenten uit de topologie ben ik een overzicht gaan maken van de diverse systeemeisen die gesteld worden aan deze componenten. Voor iedere computer in het systeem heb ik een overzicht gemaakt van welke hardware en software minimaal benodigd is en welke project software (APM server en APM client software) op welke computer zal gaan draaien.

7.3 Uitwerken logical view

De logical view documenteert het ontwerp model, waarin de lagen binnen de applicatie worden gedefinieerd. De systeem architect identificeert patronen in de functionaliteit en creëert mechanismen die in deze functionaliteit voorzien in verschillende onderdelen van de applicatie.

Allereerst ben ik begonnen met het definiëren van de verschillende lagen. Logisch gezien zal APM bestaan uit de volgende drie lagen:

1. **Userinterface laag**: Deze laag is verantwoordelijk voor het bieden van een interface aan de gebruiker en daarnaast zorgt deze layer voor de inhoud van de menu's. Data die ingevoerd wordt wordt zowel lokaal (tijdelijk, totdat alles wordt weggeschreven naar de database) als in de database opgeslagen. Tevens is deze laag verantwoordelijk voor de authenticatie en autorisatie.
2. **Business laag**: Deze laag is verantwoordelijk voor het bieden van een abstracte API die de doelen van zijn clients dient. De business laag (ook wel middelware layer genoemd) beschrijft de logica achter de systeemprocessen. Deze laag zorgt voor de interactie met andere middelware technologieën zoals een DBMS of units.
3. **Data laag**: Deze laag is verantwoordelijk voor het beheren van de data, waarbij rekening gehouden moet worden met de transacties en de consistentie van de data. De data laag brengt in kaart welke objecten in het systeem waar fysiek zijn opgeslagen (in welke tabellen).

In de logical view worden ook de stacks beschreven die door de userinterface onderhouden en gebruikt worden. Echter ik heb geen stacks beschreven omdat ik in de userinterface van APM geen gebruik zal gaan maken van stacks. Door het feit dat er in ieder geval geen sprake is van inlog procedures vervalt de stack welke account gegevens e.d. bijhoudt. Daarnaast wordt het gebruik van APM ook niet afgehandeld in de vorm van sessies. Er kan altijd 'ingeprikt' worden om de huidige unit statussen uit te lezen.

Tevens beschrijft de logical view de zogenaamde cached application reference objects. Dit zijn de objecten welke in principe zo minimaal mogelijk uit de database opgehaald worden. Binnen APM heb ik de objecten van het type unittype (het type

wat beschrijft wat de primaire taak van een unit is, bijvoorbeeld: Ad-Insertion op Eurosport of Ad-Insertion op Discovery) geclassificeerd als cached application reference objects. Deze objecten zullen namelijk slechts éénmaal, iedere keer wanneer er units of groepen worden toegevoegd, gewijzigd of verwijderd in het systeem in het geheugen geladen worden. Ik heb hiervoor gekozen, omdat deze objecten slechts zeldzaam zullen veranderen en naar verwachting nooit op het moment dat er mutaties plaats vinden op units of groepen.

7.4 Uitwerken data view

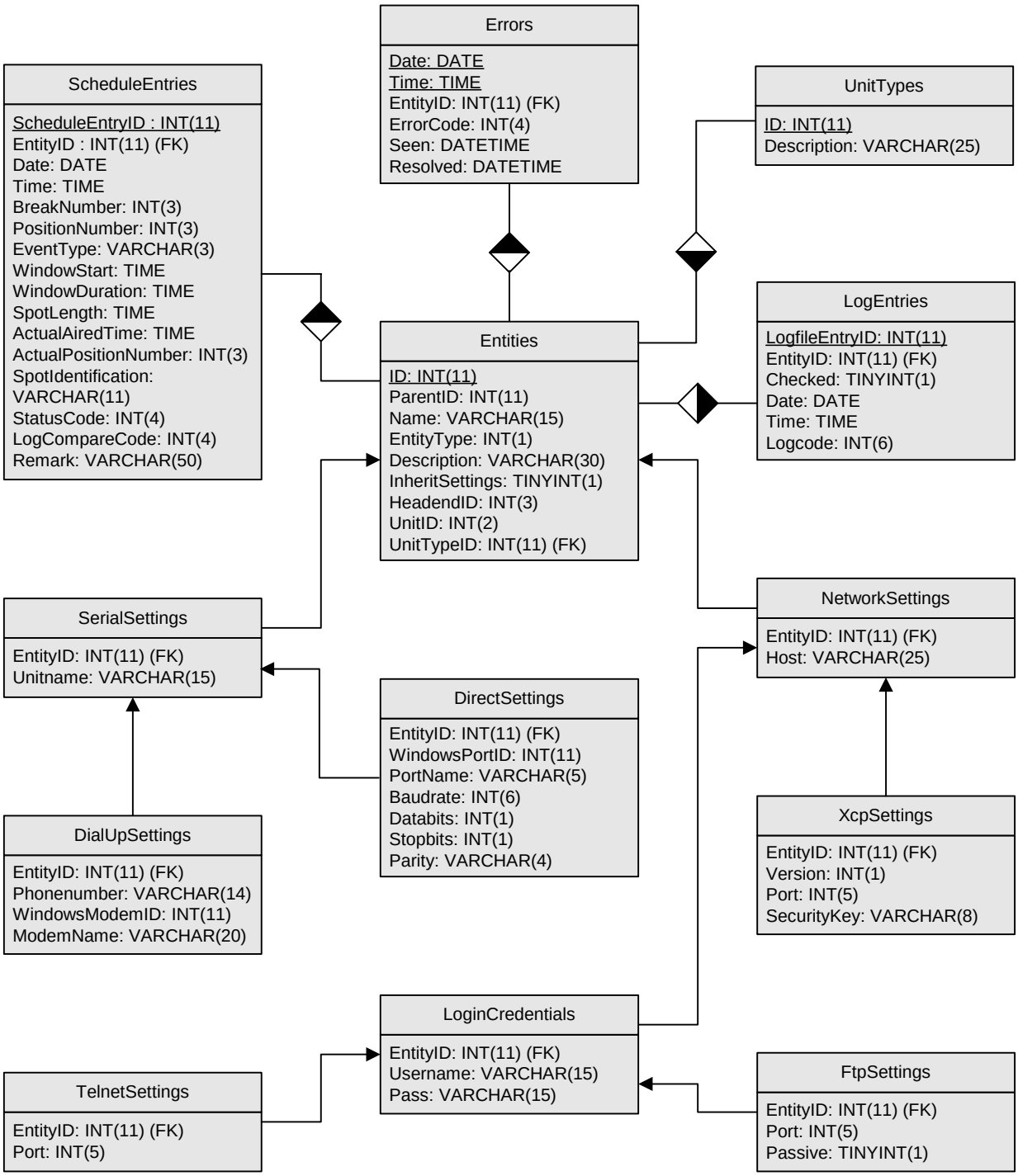
Ik heb de klassen uit het klassendiagram geclassificeerd als zijnde transient of persistent. De persistente klassen worden in kaart gebracht in structuren op disk, meestal in een combinatie van rijen in een relationele database. Een entity-relationship diagram beschrijft het database schema. Deze view brengt ook in kaart hoe de objectgeoriënteerde klassen in de relationele tabellen staan.

In de figuur op de volgende bladzijde staat het ERD zoals ik het gemaakt heb afgebeeld. Tenopzichte van het oorspronkelijke klasse diagram is het voor een groot deel een één-op-één vertaling hiervan. Er is duidelijk te zien dat veel van de klassen overgenomen zijn als tabel.

De klassen ‘LogFiles’ en ‘Schedules’ zijn niet overgenomen als tabel. Dit heb ik gedaan, omdat aparte tabellen voor deze gegevens niet veel zouden toevoegen. Het enige attribuut wat deze beide klassen hadden was ‘Date’. Echter, dit attribuut zou ook terug moeten komen bij de ‘entries’ tabel van deze beide klassen, om te verwijzen naar de schedule- of logfile waar het deel van zou zijn. Het zou dus op geen enkele wijze winst opleveren.

Een ander verschil is dat de attributen die bij het klasse diagram niet volledig ingevuld waren bij het ERD wel volledig opgenomen zijn. Dit om een zo volledig mogelijk beeld van de data die opgeslagen moet worden te geven.

Daarnaast heb ik het atribuut ‘UnitType’ uit de klasse ‘Entities’ een eigen tabel gegeven zodat vergelijking op UnitTypeID mogelijk is. UnitType is de tabel die de verschillende soorten Media Choice units definieert (Ad-Insertion Eurosport, Ad-Insertion Discovery, etc.)



Figuur 7-2 Logical View op de database

Aan de hand van het ERD heb ik de create queries van de database opgesteld. Deze kunnen gebruikt worden om de database daadwerkelijk te maken.

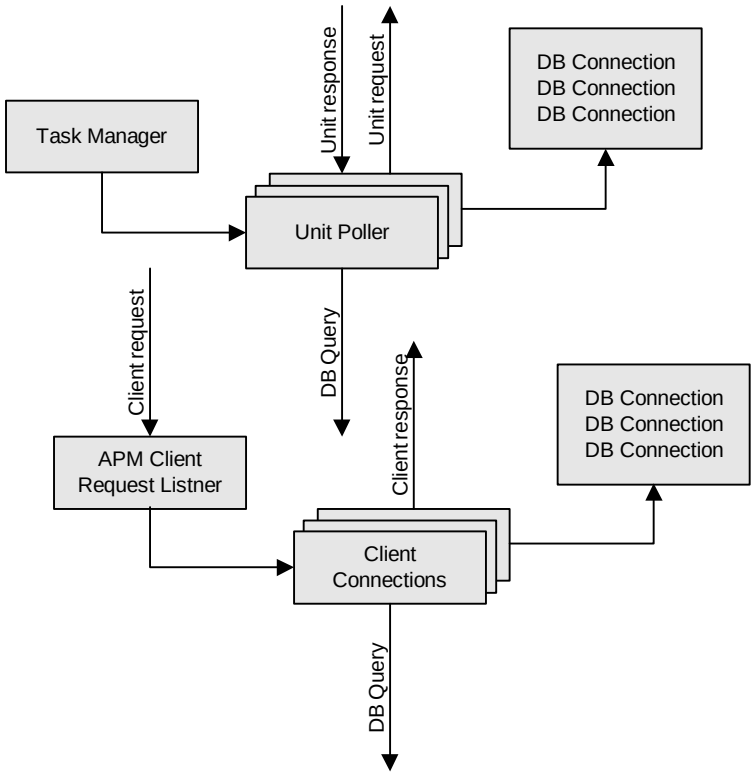
7.5 Uitwerken process (concurrency) view

Deze view richt zich het concurrency aspect van het systeem en hoe er wordt omgegaan met gedeelde bronnen/hulpmiddelen. De process view documenteert de onafhankelijke communicatiesystemen binnen het systeem en beschrijft ook hoe

deze communiceren. Daarnaast geeft het een overzicht van de bronnen die door deze lijnen worden gebruikt en het transactiemodel dat wordt gebruikt voor het behouden van de integriteit van deze bronnen. Binnen APM heb ik twee verschillende vormen van concurrency onderscheiden:

- CPU/process/threads ontwerp
- Database transacties

In het CPU ontwerp heb ik een schets gemaakt van de threads die in APM gecreëerd zullen worden. Dit heb ik gedaan om een beeld te vormen van de verschillende soorten threads die ik in APM dien te implementeren. Ik heb deze twee soorten threads onderscheiden, omdat dit de twee taken zijn die meerdere malen tegelijkertijd uitgevoerd moeten kunnen worden. De unit poller, omdat die meerdere units tegelijkertijd uit moet kunnen lezen en de APM client connector (zoals de APM Client Request Listner met de Client Connections samen genoemd worden), omdat die meerdere APM clients tegelijkertijd moet kunnen bedienen.



Figuur 7-3 APM Server concurrency

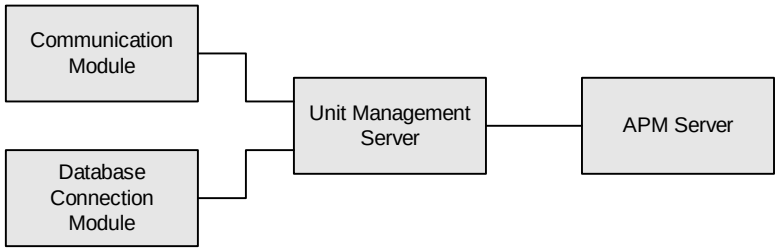
Voor de database transacties heb ik beschreven hoe de transacties vanuit de business laag (de laag die contact onderhoud met andere systemen) aangeroepen worden en data retourneren. In principe zal de task manager in APM Server er voor zorgen dat alle threads unieke en relevante data opvragen bij de units en invoeren in de database. Voor de zekerheid zal er ook gebruik gemaakt worden van optimistic

concurrency om de data-integriteit te waarborgen. Bij iedere update wordt allereerst gecontroleerd of de in te voeren data daadwerkelijk nieuwe data betreft. Dit gebeurd door eerst te controleren wat de datum en tijd van de laatst ingevoerde logentries is en deze te vergelijken met de datum en tijd van de nieuw in te voeren logentries.

7.6 Uitwerken implementation view

Binnen deze view wordt aangegeven welke klassen horen bij welke fysieke bronbestanden en combineert de bestanden in werkbare componenten. De Implementation view houd ook bij welke afhankelijkheden er bestaan tussen componenten.

Ik heb APM globaal opgedeeld in twee gedeeltes, een server en een client gedeelte. De client zal bestaan uit één geheel, de server zal bestaan uit meerdere gedeeltes van herbruikbare componenten.



Figuur 7-4 Server Implementatie model

Ik heb gekozen voor meerdere componenten, omdat de verwachting bestaat dat in de toekomst het systeem uitgebreid zal worden met nieuwe diensten. De Unit Management Server fungeert in dit geheel dan als component wat alle diensten (zoals APM Server) in één systeem integreert. De diensten kunnen gebruik maken van de modules van de unit management server om verbinding te maken met units en databases. Deze modules zijn opgesplitst, omdat in de toekomst deze mogelijk vervangen, verwijderd of uitgebreid kunnen worden met nieuwe of andere functionaliteiten. Mogelijke veranderingen in de modules kunnen zijn: updates in het XCP protocol, de keuze voor een ander DBMS, etc.

Na deze opdeling heb ik de diverse onderdelen gekoppeld aan de klassen waarmee ze zullen werken en onderverdeeld in de drie gedefinieerde software lagen

UI laag:

- APM Client (Errors, Entities)

Business laag:

- APM Server (Schedules, ScheduleEntries, LogFiles, LogEntries, Errors)
- Unit Management Server (Entities)

- Communication Module (SerialSettings, NetworkSettings, DialUpSettings, DirectSettings, LoginCredentials, XcpSettings, TelnetSettings, FtpSettings)
- Database Communication Module (Schedules, ScheduleEntries, LogFiles, LogEntries, Erros, Entities, SerialSettings, NetworkSettings, DialUpSettings, DirectSettings, LoginCredentials, XcpSettings, TelnetSettings, FtpSettings)

Data laag:

- Database tabellen
- Database Connection Module

7.7 Uitwerken security view

Deze view richt zich op hoe gebruikers zich identificeren, hoe rechten worden toegekend op basis van identiteit, hoe zorg wordt gedragen voor de integriteit van het systeem en de data.

Omdat APM slechts minimaal gebruik zal maken van beveiliging heb ik over dit onderwerp slechts een punt kunnen beschrijven, namelijk de data integriteit en privacy.

Paswoorden die gebruikt worden om contact te maken met de units zullen worden versleuteld voordat ze worden opgeslagen in de database, dit omdat bij een eventuele inbraak op de database van APM dan de connectie gegevens van de units niet direct beschikbaar zijn, zodat op het playout proces van de units geen invloed uitgeoefend kan worden. Tevens wordt het paswoord wat wordt gebruikt om verbinding te maken met de database encrypted opgeslagen in het registery, dit is om te voorkomen dat iemand die inbreekt op APM server niet ook op de database in kan breken.

8 MAKEN VAN HET TESTPLAN

In het testplan heb ik beschreven hoe het uiteindelijke systeem getest gaat worden. Het opstellen van het testplan heb ik gedaan aan de hand van drie onderdelen:

1. **Test strategieën**: Definieert de diverse tests;
2. **Test set ontwerpen**: Definieert de diverse test sets;
3. **Test sets**: Beschrijft de diverse tests.

8.1 Test strategieën

Het eerste stadium wat ik heb doorlopen in het opstellen van het test proces van het systeem is het definiëren van een test strategie. Een test strategie beschrijft de globale test wijze, identificeert de verschillende benodigde test niveaus en de methodes, technieken en gereedschappen die benodigd zijn. Het hart van de strategie is het analyseren van de eisen en het identificeren van de test sets, welke gezamenlijk controleren of het systeem aan alle systeem eisen voldoet. De strategie beschrijft ook de te gebruiken test data.

Ik heb ervoor gekozen de volgende testen uit te voeren:

- **Procedure/functie test**: Controleer de werking van een procedure of functie;
- **Module test**: Controleer de werking van een module;
- **Integratie test**: Combineer twee modules en controleer de werking;
- **Systeem test**: Controleer de werking van het gehele systeem.

Op deze manier kan ik van onderaf controleren of alles functioneert zoals het hoort. Door te beginnen bij het testen van de procedures en op te klimmen naar het testen van de modules (verzameling procedures en functies die met elkaar één taak uitvoeren) kan ik de testen steeds breder maken. Bij de integratie test kan ik dan controleren of modules met elkaar kunnen werken om vervolgens in de systeem test het volledige systeem te controleren op haar werking.

Voor het testen van het systeem heb ik het volgende stappenplan gevolgd om de systeem testen te ontwikkelen:

Stap 1: Analyseer alle systeem eisen;
Stap 2: Benader de systeem risico's vanuit een systeem test perspectief;
Stap 3: Definieer de test sets, gebruikmakend van de systeem risico's;
Stap 4: Leg de systeem eisen naast de test sets en sla dit op. Zorg dat de test sets bijgehouden worden wanneer het systeem evolueert;
Stap 5: Ontwerp de test cases per test set;
Stap 6: Creëer de test cases voor iedere test set.

Stap 4 t/m 6 heb ik bij het maken van het testplan niet uitgevoerd. Deze zullen gedaan worden op het moment dat het systeem ook daadwerkelijk geïmplementeerd wordt.

Door het volgen van stap 1 t/m 2 heb ik lijst opgesteld met mogelijke risico's, aannames en beperkingen.

8.2 Test sets ontwerpen en beschrijven

Bij stap 3 heb ik drie test sets gedefinieerd waarin het systeem op alle door mij gedefinieerde risico's getest werd. Deze test sets zijn:

1. Core Functionaliteit/Belasting/Concurrency Test Set
2. Beveiliging/Beschikbaarheid Test Set
3. Gedetailleerde Test Set

Core Functionaliteit/Belasting/Concurrency test set: Deze test set verifieert dat APM het bedrijfskundige probleem wat het op moet lossen ook daadwerkelijk oplost. De units van Media Choice worden 24/7 gemonitord en problemen gedetecteerd en gerapporteerd.

Beveiliging/Beschikbaarheid test set: Deze test set verifieert dat APM voldoende beveiligd is tegen niet toegestane verbindingen en het verifieert dat APM robuust genoeg is voor de gewenste 24/7 beschikbaarheid.

Gedetailleerde test set: Deze test set verifieert dat alle gedefinieerde details ook daadwerkelijk geïmplementeerd zijn in het systeem. Daarnaast verifieert het ook dat alle functionaliteiten naar behoren werken.

8.3 Acceptatie test

Het testplan document heb ik afgesloten met een beschrijving van de acceptatie test van APM door Media Choice. Media Choice zal APM accepteren als alle systeem testen succesvol verlopen en geen kritische fouten optreden. Hoge prioriteitsfouten mogen optreden mits dit een verwaarloosbaar aantal is en hier zogenaamde work-arounds voor gemaakt kunnen worden.

9 REALISATIE IN BORLAND DELPHI 7

9.1 De ontwikkelomgeving

Delphi 7 is een goed gestructureerde ontwikkelomgeving die, in de verte, wel iets weg heeft van J-Builder, het ontwikkel pakket wat ik heb leren gebruiken op de Haagse Hogeschool. Delphi is een 4GL programmeertaal. Boven in het scherm staan de controls welke op formulieren geplaatst kunnen worden. Een handige functie is het dubbelklikken op een object op een formulier. Dit brengt automatisch het code venster naar de voorgrond en maakt gelijk een procedure aan voor het meest voorkomende event van de control waarop geklikt werd.

De programmeertaal Delphi is afgeleid van de taal Pascal, het wordt ook wel Object Pascal genoemd. Ik ben van mening dat alle moderne 4GL programmeertalen en omgevingen in essentie erg veel op elkaar lijken (Visual C, Visual Basic, Delphi, C++ Builder, etc.), vandaar dat het voor mij ook geen groot probleem was om in Delphi aan de gang te gaan. Ik heb in het verleden met diverse talen ervaring opgedaan waaronder Pascal waar Delphi opgebaseerd is.

9.2 Het maken van de userinterfaces

Het plaatsen van controls op formulieren is in Delphi zeer eenvoudig te doen. Echter, het is wel zaak dat er nagedacht wordt over het doel van een scherm. Reeds vanaf de start van de opdracht had de opdrachtgever een duidelijk beeld van hoe de schermen moesten worden. In overleg met hem heb ik dan ook de eerste versies van de prototypes van deze schermen in Delphi geïmplementeerd. Vervolgens heb ik samen met een duaal student VIA die bij Media Choice werkt de schermen zo duidelijk mogelijk gemaakt. Aan de hand van deze prototypes ben ik gaan ontwikkelen.

9.3 Modulair opbouwen van het pakket

Omdat de gehele applicatie zo modulair mogelijk opgebouwd diende te worden heb ik onderzoek verricht op internet naar DLL’s (Dynamic Link Libraries). Dit zijn aparte executables die een verzameling functies en/of procedures bevatten welke gebruikt kunnen worden in applicaties. DLL’s zijn niet zonder een zogenaamde host applicatie uit te voeren. Door het uiteindelijke pakket op te bouwen met behulp van DLL’s is het relatief simpel om functies her te gebruiken, te vervangen of uit te breiden zonder het gehele pakket onderhanden te hoeven nemen.

Aan de hand van de handleidingen die ik op internet vond ben ik begonnen met het maken van een test applicatie die één functie uitvoerde welke uit een zelf gemaakte DLL kwam. Dit bleek relatief eenvoudig. Hierna heb ik zelf nog een aantal procedures en functies geschreven en toegevoegd aan de DLL, dit werkte allemaal zonder al te veel problemen. Deze test DLL is vervolgens uitgegroeid tot de basiscode voor de DLL’s die ik daarna zou schrijven.

9.4 Maken van de database

Binnen Media Choice wordt voor meerder toepassingen gebruik gemaakt van een MySQL database server, vandaar dat ook voor APM een MySQL server ter beschikking stond.

Reeds in het Software Architecture Document had ik een ERD gemaakt van de te gebruiken database. Naast deze ERD had ik ook de create codes opgesteld. Aan de hand van deze create codes heb ik de database via de MySQL console applicatie gecreëerd.

9.5 Verbinding leggen met de database server

9.5.1 ADO en BDE

In Delphi zit standaard een aantal componenten waarmee het mogelijk is om verbinding te maken met een database server. Het is mogelijk om op een tweetal manieren een verbinding te maken, via ADO (ActiveX Data Objects) of via BDE (Borland Database Engine). ADO is een methode om met databases te communiceren die standaard in Windows is geïntegreerd. BDE daarentegen is een specifiek component voor applicaties die ontwikkeld zijn met ontwikkelomgevingen van Borland (de maker van o.a. Delphi). BDE vereist dan ook dat het met iedere applicatie die hiervan gebruik maakt mee gedistribueerd wordt aan klanten.

Via zowel ADO als BDE is het mogelijk om een verbinding te maken met een via ODBC (Open Database Connectivity) verbonden database. In tegenstelling tot ADO en BDE bieden de meeste database systemen wel ondersteuning voor ODBC. Voor MySQL bestaat er dus ook een ODBC driver.

Ik heb gekozen om gebruik te maken van ADO. Dit is voor een deel een praktische keuze, het mee distribueren van BDE vind ik overdreven als iets soortgelijks in Windows is geïntegreerd, en een keuze om de continuïteit te waarborgen. Na een onderzoek op internet over dit onderwerp las ik op veel plaatsen dat ervaren software ontwikkelaars BDE achterhaalt vonden en verwachten dat deze niet lang meer ondersteund zou worden. Als dat daadwerkelijk zou gebeuren zou er ook geen ontwikkeling meer op plaats vinden. Dit zou betekenen, indien er nog fouten in worden ontdekt dat APM daar wel last van zou kunnen ondervinden zonder dat de kans bestaat dat deze ooit opgelost worden.

9.5.2 Ontwikkelen van de database connector

Op basis van de test DLL die ik in de beginfase van de ontwikkeling heb geschreven heb ik een DLL gemaakt die queries uitvoerde op een in het registry van Windows gedefinieerde database.

Hierbij stuitte ik echter nog wel op een probleem. In eerste instantie had ik een applicatie gemaakt met een formulier. Op dit formulier had ik de componenten waarmee de database verbinding opgezet werd geplaatst. Echter deze componenten zijn zogenaamde VCL's (Visual Component Libraries). Zoals de naam al zegt zijn deze componenten visueel en hebben dus een formulier nodig om te bestaan. Ze maken het mogelijk om zonder enige regel code te schrijven een verbinding te maken en een query uit te voeren. Deze componenten zijn ook zonder formulier te gebruiken, maar dan moeten ze handmatig in de code aangemaakt worden. Dit vereiste enig onderzoek, maar is uiteindelijk wel gelukt.

Uiteindelijk heb ik een DLL op kunnen leveren die een enkele functie uitvoerde die het mogelijk maakt om vanuit de host applicatie een query uit te voeren en de response van de database te retourneren.

9.6 Communicatie met de units

Communicatie met de units kan op verschillende manieren geschieden. Allereerst is er de mogelijkheid van telnet en FTP. Via het telnet protocol kan er ingelogd worden op de units en commando's uitgevoerd worden. Via FTP is het mogelijk bestandsbeheer te plegen op de units. Daarnaast bestaat er het RS232 protocol waarmee er direct, of via een modem, commando's uitgevoerd kunnen worden. Het laatste protocol wat de units ondersteunen is het eigen XCP protocol.

9.6.1 Communicatie via XCP

XCP (X Command Protocol) is een door Adtec Inc. ontwikkelt protocol op basis van UDP. Dit protocol was in eerste instantie ontwikkeld voor communicatie tussen units onderling, echter de technische medewerkers bij Media Choice zagen dit protocol als een goede optie om snel en eenvoudig status data uit te lezen. Vandaar dat ik de opdracht kreeg om, als onderdeel van het totale pakket, een implementatie van dit protocol te schrijven.

Vanuit Adtec Inc. was er wel documentatie beschikbaar over XCP, maar deze was in eerste instantie geschreven als korte beschrijving voor eigen engineers. Tevens was deze beschrijving geschreven met XCP versie 1 in gedachten, maar ondertussen bestond versie 2 al. Met deze handleiding en een zogenaamde packet sniffer ben ik data die tussen units heen en weer werd verzonden gaan analyseren en heb ik het XCP versie 1 en 2 protocol weten te ontcijferen. Uiteindelijk heb ik een XCP encoder en decoder geschreven. Met deze applicatie kon ik unit commando's encoderen naar XCP data en XCP data decoderen naar unit commando's.

Tijdens het uitzoeken van de XCP structuur heb ik een probleem in het protocol ontdekt. Het protocol, zoals tot dan toe geïmplementeerd, maakte altijd gebruik van poort 44527, zowel voor het ontvangen als voor het versturen van de data. In het TCP/IP protocol (waar UDP, en dus XCP, toe behoort) is het geregeld dat in een client/server architectuur de client zelf bepaalt op welke poort data uitgestuurd en

ontvangen wordt. Met de huidige XCP implementatie op de units was dat niet mogelijk. Alle data zou altijd op poort 44527 terug gestuurd worden naar de client. In de praktijk zou dit dus betekenen dat slechts één client applicatie of thread op een PC gebruik zou kunnen maken van XCP. Omdat de toekomstige applicatie multi-threaded opgebouwd zou gaan worden was dit een probleem. Tevens was het, zo bleek uit testen, een probleem bij diverse routers. Aangezien de, van XCP gebruik makende applicatie, zich gedroeg als server (vaste poort voor inkomende data) en routers daar in de regel apart voor geconfigureerd moeten worden. Door mijn bedrijfsmentor is dit probleem aangekaart bij de engineers van Adtec Inc. Een tweetal maanden later kreeg hij bericht van deze organisatie dat deze gang van zaken in de units inderdaad tot problemen kon leiden en dat er om die reden een nieuwe versie van het protocol was ontwikkeld, versie 3. Units gebaseerd op XCP versie 3 luisterden wel op poort 44527, maar stuurden data terug op de door de client geopende poort.

Ook dit XCP component heb ik uiteindelijk, met behulp van de studie DLL, in DLL vorm gegoten.

9.7 Controleren van playout

Een zeer belangrijk onderdeel van APM is het gedeelte wat controleert of de unit alle ingeplande spots ook daadwerkelijk heeft uitgespeeld.

9.7.1 Playout controle aan de hand van log gegevens

Hiervoor heb ik een procedure geschreven die de database raadpleegt en de scheduleentries van een bepaalde dag van een bepaalde unit en de, onbehandelde, logentries van diezelfde dag en unit ophaalt. Vervolgens gaat de procedure ieder logitem af. Allereerst wordt de logcode gelezen. Indien deze 2000 (Play) is dan zal het logitem behandelt worden, anders wordt er doorgegaan naar het volgende item. Wanneer een logitem behandelt wordt zal in de scheduleentries de spot gezocht worden die dezelfde naam draagt en waar het Time attribuut van de logentries tussen WindowStart en WindowStart + WindowDuration ligt. Indien deze bestaat zal bij het betreffende ScheduleEntry het attribuut ActualAiredTime gevuld worden met Time uit het betreffende LogEntry. Daarna zal de ingeplande spot van die dag met dezelfde WindowStart en WindowDuration en de hoogste ActualPositionNumber opgevraagd worden. Vervolgens zal ActualPositionNumber van de uitgespeelde spot gevuld worden met het opgevraagde ActualPositionNumber opgehoogd met 1. Uiteindelijk zal LogCompareCode van de uitgespeelde spot gevuld worden met 0001 (Played in log)

Indien een spot volgens de logentries wel is uitgespeeld, maar niet terug gevonden kon worden in de scheduleentries zal de tabel Errors gevuld worden. Een nieuw record zal gemaakt worden. Deze krijgt de Date en Time gelijk aan de gelijknamige velden uit het bewuste LogEntry. Vervolgens zal een Errorcode 2000 (Unscheduled spot played) meegegeven worden.

Wanneer het in de ScheduleEntries ActualPositionNumber niet gelijk is aan PositionNumber wordt de Errors tabel ook gevuld. Een nieuw record met Errorcode 4000 (Other position number) zal gemaakt worden.

Bij alle logentries die behandelt zijn wordt de boolean waarde Checked op TRUE gezet.

Wanneer alle beschikbare log gegevens verwerkt zijn worden alle ScheduleEntries opgevraagd van de laatste twee dagen waarbij WindowStart + WindowDuration voor de laatst behandelde logentry Date en Time ligt en de LogCompareCode op 0000 (Unchecked) staat. Indien er records retour komen blijken deze niet uitgespeeld te zijn. Deze krijgen dan de LogCompareCode 0002 (Did not play in log). Vervolgens wordt gecontroleerd of het om één of meerdere spots gaat of om een geheel ad-insertion blok. Dit gebeurt door alle scheduleentries met dezelfde WindowStart en WindowDuration als van de niet gespeelde spot op te vragen. Indien hier spots voorkomen met LogErrorCode 0001 (Played in log) gaat het niet om een geheel blok. De tabel errors zal dan gevuld worden met een nieuw record met de Date en Time attributen van de bewuste scheduleentry en de ErrorCode 1001 (Missed Spot). Als alle ScheduleEntries met dezelfde WindowStart en WindowLength de waarde 0002 hebben zal de tabel Errors met een record gevuld worden met dezelfde Date en Time van de spot met de laagste PositionNumber en met ErrorCode 1000 (Missed insert).

10 OPLEVEREN APPLICATIE

Vanwege het feit dat de applicatie niet volledig is afgerond en ik bij het afsluiten van de afstudeerperiode nog volop in het ontwikkelproces was verwikkeld ben ik niet aan het samenstellen van een beknopte handleiding (quick reference card) toe gekomen. Hierdoor heeft ook de uiteindelijke oplevering van een installatiepakket en het uitvoeren van een acceptatietest nog geen doorgang kunnen vinden.

11 EVALUATIE

Het einde van het afstuderen is nabij en deze gehele periode dient geëvalueerd te worden. In dit hoofdstuk heb ik een terugblik op het verloop van het ontwikkeltraject beschreven. Daarnaast zijn de procesgang en de opgeleverde producten geëvalueerd.

11.1 Proces evaluatie

Het lijkt aan de éne kant alsof de tijd voorbij gevlogen is en lijkt het alsof ik nog lang geen twintig weken bij Media Choice aan de gang ben geweest. Aan de andere kant lijkt het echter een eeuwigheid geleden dat ik de eerste letters van de opdrachtomschrijving op papier zette. Hieronder heb ik de procesgang geëvalueerd per deelopdracht die ik gedurende het afstuderen heb uitgevoerd.

11.1.1 Eerste gedachten over de opdracht

Toen de opdracht mij werd aangeboden sprak deze mij meteen al bijzonder aan. Voornamelijk het feit dat ik in staat gesteld werd om software met een zekere mate van intelligentie te ontwikkelen, iets wat ik tot dan toe nog niet gedaan had, vond ik erg interessant. Ten tweede leek het feit dat Media Choice bezig is met, voor Europese begrippen, nieuwe diensten en dat ik daar in mee kon helpen voor mij een ontzettend leuke bijkomstigheid. Daarnaast sprak het mij ook technisch bijzonder aan. De apparatuur waarmee gewerkt mocht gaan worden, de Adtec MPEG decoders, zijn zeer geavanceerde videospelers.

11.1.2 Aanvang van de opdracht

Alvorens ik daadwerkelijk aan de opdracht kon beginnen diende ik eerst wat kennis op te doen betreffende de organisatie, RUP en de MPEG decoders welke de applicatie moest gaan monitoren.

In eerste instantie zag ik op tegen de communicatie met de unit. De opdrachtgever en bedrijfsmentor wilden graag dat ik gebruik zou gaan maken van het XCP protocol, maar konden mij niet verder helpen dan de summiere documentatie die zij van de fabrikant van de units hadden ontvangen. Mede hierom ben ik al in een vroeg stadium begonnen met onderzoek te verrichten naar dit protocol. Na, het zogenaamde, port-sniffen en het analyseren van verzonden en ontvangen pakketten kreeg ik na enige tijd, in combinatie met de documentatie, enig zicht op de opbouw en werking van dit protocol.

Voordat ik aan de slag kon moest ik, zoals al eerder vermeld, eerst mijn eigen werkstation configureren. Dit configureren ging zonder problemen. Mijn ervaring op dit gebied in mijn privé leven en bij mijn vorige werkgever kwamen goed van pas.

11.1.3 Plan van Aanpak

Het plan van aanpak opstellen heeft geen grote problemen opgeleverd. Mede door de ervaring die ik op dit gebied ruimschoots heb opgedaan op school was deze

relatief snel opgesteld. Helaas is achteraf de planning niet helemaal correct gebleken. Maar ook uit ervaring heb ik geleerd dat dit geen vreemd fenomeen is.

11.1.4 Business Architecture document

Het Business Architecture document was het eerste RUP document wat ik heb gemaakt. Op internet had ik, al voor aanvang van het afstuderen, een voorbeeld project gevonden wat volledig met RUP ontwikkeld was. Aan de hand van het Business Architecture document wat daarbij zat ben ik een Business Architecture document voor APM op gaan stellen. Dit was een vrij voorspoedig lopend proces. Omdat ik een redelijk goed beeld had van de organisatie en de gewenste applicatie kon ik vrij snel alles op papier zetten.

Het Business Architecture document heb ik met mijn bedrijfsmentor doorgenomen. Hij vond het aantal use cases vrij weinig, maar na enige discussie kwamen we tot de conclusie dat er niet veel use cases zijn voor APM en dat het Business Architecture document sowieso nog niet veel diepgang hoeft te hebben. Daarnaast hebben we ook, zoals ik ook beschreven had bij de proces beschrijving, een punt van discussie gehad met betrekking tot het klasse diagram.

11.1.5 Requirements document

Na nog wat kleine aanpassingen gemaakt te hebben aan het Business Architecture document kon ik mij gaan richten op het maken van een Requirements document. Er ging weliswaar redelijk veel tijd in zitten, maar voor veel problemen heeft het niet gezorgd. Ook bij dit document heb ik mij voor de structuur laten beïnvloeden door het voorbeeld RUP project wat ik had, in dit geval door het daarbij horende Requirements Document. Één probleem waar ik op stuitte was het Use Case Object Model. Ik zag geen mogelijkheid om in dit object model meer diepgang, dan het toevoegen van een extra actor, te brengen, terwijl ik wel de indruk had dat ik dit zou moeten doen in het Requirements Document. Ik kon wel de objecten uitbreiden met alle gegevens die in een logfile of schedulefile voorkomen, maar dit heb ik niet gedaan, omdat het object model hier alleen maar voller en minder overzichtelijk van zou worden terwijl het weinig tot niets toe zou voegen.

Veel problemen leverde het Requirements Document verder niet op. Wel nam het meer van de tijd in beslag dan ik ervoor had gereserveerd.

11.1.6 Software Architecture document

Uit het voorbeeld Software Architecture document kon ik, net als bij de voorgaande documenten, gelukkig gebruik maken van de hoofdstukken structuur.

De eerste paar hoofdstukken waren redelijk probleemloos op te stellen. Echter bij het beschrijven van de Logical View liep ik enigszins vast. In het voorbeeld document werd gesproken over de userinterface session stack. Enerzijds kon ik geen userinterface session stack voor APM opstellen, anderzijds leek het me in eerste instantie vrij onwaarschijnlijk dat er geen sprake van zou zijn in APM. Hetzelfde gold

voor de scenario's. Op dat moment ben ik de twee projecten eens naast elkaar gaan leggen. In het voorbeeld document gingen ze uit van een webwinkel. Een webwinkel bestaat uit vrij veel verschillende formulieren en moet er tussen verschillende formulieren relatief veel data uitgewisseld worden. Binnen APM was er van dit soort zaken geen sprake. De uiteindelijke userinterface bestaat uit twee opzichzelfstaande formulieren. Op basis van deze vergelijking kwam ik tot de conclusie dat APM inderdaad geen userinterface stack gebruikte en dat ook het aantal scenario's niet in de buurt kwam van het aantal scenario's van een webwinkel.

De overige hoofdstukken waren, nadat ik de verschillen tussen de twee projecten voor mezelf op een rijtje had gezet, ook zonder al teveel problemen te doen. Met name de Data View was vrij snel in elkaar gezet, omdat ik onbewust het initiële klasse diagram al vanuit een implementatie perspectief had opgezet. Het ERD was dus een bijna directe vertaling van het klasse diagram

11.1.7 Testplan

Het testplan is een vrij klein, maar niet onbelangrijk document. Opstellen hiervan ging vrij voorspoedig. Met behulp van het zesstappenplan wat ik uit het voorbeeld testplan document had overgenomen kon ik het testplan in de geplande tijd opstellen. Het enige puntje van het testplan waar ik nog enige hulp bij in heb geroepen van mijn bedrijfsmentor is het bepalen van de risico's die gelopen worden bij het gebruik van APM. Hierdoor duurde het opstellen van dit hoofdstuk vrij lang, maar ik ben in de overtuiging dat we daardoor wel bijna alle reële risico's hebben kunnen definiëren.

11.1.8 Ontwikkelen in Delphi 7

Bij het programmeren ben ik begonnen met alle functies en procedures die in de business laag liggen. Ik heb hiervoor gekozen, omdat deze zaken de basis vormen van de gehele applicatie. Daarnaast zaten in deze onderdelen, op het eerste gezicht, de voor mij de twee grootste uitdagingen. Namelijk het implementeren van de intelligentie en het schrijven van de communicatie module met de unit.

Beide functies hebben mij dan ook de nodige tijd werk gekost. Hoewel ik reeds in een vroeg stadium de structuur van het XCP protocol had uitgezocht, bleek de implementatie hiervan niet direct simpel. Veel bladeren in mijn Delphi boek en zoeken op internet waren het gevolg, maar uiteindelijk met een werkend eindresultaat. De hieruit afkomstige DLL heb ik ook al met goed gevolg in kunnen zetten in een klein project wat ik tussen de ontwikkeling door heb uitgevoerd voor Media Choice.

Voor de ontwikkeling van de play-out controle functies heb ik eerst een schets gemaakt waarin ik de doorgang van de functies beschreef. De module werd namelijk door de verschillende fouten waarop getest wordt en de wijze waarop deze getest worden vrij complex. Met behulp van dit diagram en door stapsgewijs op te bouwen verliep de ontwikkeling redelijk gemakkelijk. Echter, eigenlijk had dit diagram, wat

een beschrijving van de implementatie inhield, thuis gehoord in het Software Architecture document.

11.1.9 Het gebruik van RUP

Het starten van het project was voor mij een punt waar ik enigszins tegenop zag. Voornamelijk het feit dat ik geen enkele kennis had van RUP zorgde daarvoor. Vandaar dat ik al voordat met het afstuderen begon enige onderzoek op dit gebied heb gedaan.

Een groot gedeelte van de kracht van RUP ligt in het feit dat het een project beheersmethode is die volledig flexibel is, ieder project kan zelf bepalen welke artefacten nodig zijn. Behalve dat dit een grote kracht is, was het voor mij ook een groot struikelblok. Vooraf vond ik het heel erg moeilijk te bepalen hoe ik het aan moest pakken, welke artefacten wel te maken en welke niet.

Zoals ik al eerder beschreef had ik de beschikking over een ander project wat met RUP was uitgevoerd. Dit was de leidraad die ik uiteindelijk gevolgd heb. Daarnaast heb ik mij uiteraard nog wel verdiept in literatuur en artikelen over dit onderwerp.

Hoewel RUP een van oorsprong iteratieve ontwikkelmethode is, heb ik hier niet veel mee gedaan tijdens de ontwikkeling van APM. Alles is als in een waterval methode uitgerold. Ik heb wel af en toe documenten gewijzigd terwijl ik ze eigenlijk al afgerond had, maar hier hield het iteratieve dan ook wel op. Echter, de verwachting is wel, uitgaande van de uitbreidbaarheids wens van de opdrachtgever dat ik nog wel een keer gebruik zal gaan maken van het iteratieve karakter van RUP.

Daarnaast ben ik ook teveel bezig geweest om artefacten in één versie af te maken en af te sluiten. Dit is, zoals ook geschreven wordt in de door mij gebruikte literatuur, logisch, maar niet goed. Het is niet de bedoeling een artefact af te maken, maar om het zo ‘volwassen’ te maken dat er mee gewerkt kan worden en op deze basis beslissingen genomen kunnen worden. Dit is iets waar ik me niet altijd aan gehouden heb.

11.2 Product evaluatie

In deze paragraaf zijn de evaluaties opgenomen betreffende de opgeleverde producten. Er is besproken hoe de producten tot stand zijn gekomen, welke aanpassingen er na de eerste versie aan verricht zijn en, tot slot, mijn eigen mening betreffende het product.

11.2.1 Plan van aanpak

Na een eerste gesprek met de opdrachtgever, die mij de opdracht had voorgelegd en nader had toegelicht, diende ik de opdracht te omschrijven voor de examinatoren van de Haagse Hogeschool die, na een aantal kleine aanpassingen in de manier van omschrijven, de opdracht goed keurde. Deze omschrijving vormde de basis voor het

plan van aanpak. Verder bezat het plan van aanpak nog de uitgangspunten en een planning. Dit was versie 1 en de gehanteerde versie van het plan van aanpak.

Wanneer ik terug kijk op het plan van aanpak vind ik dat ik er wel tevreden over mag zijn. Toch dient de eerlijkheid mij te zeggen dat, als ik het nu opnieuw zou moeten doen, ik het anders zou doen. De tijd die ik voor de diverse onderdelen reserveerde was niet altijd reëel. Met name het opstellen van het Requirements Document, het opstellen van het Software Architecture Document en het implementeren hebben meer tijd gekost.

11.2.2 Business Architecture document

In het Business Architecture document heb ik een aantal zaken beschreven waar ik achteraf van denk dat ik het anders had moeten doen. Zo heb ik een context diagram opgenomen die zo summier is dat het niets toevoegt. Ik dacht toen ik het maakte dat het systeem zou verduidelijken, maar achteraf gezien betwijfel ik dat.

Vanwege het feit dat het systeem vrij weinig use cases kent en ik in het kader van de globaalheid niet direct alle mogelijkheden wilde presenteren zijn de use cases die ik heb opgesteld wel zo zeer eenvoudig dat ze niets toevoegen.

Bij het opstellen van het domein object model voorbeeld en het klasse diagram heb ik ook een fout gemaakt. Een docent heeft mij ooit eens verteld dat klasse diagrammen te vergelijken zijn met ERD’s. In principe is het mogelijk om een klasse diagram zonder al te veel wijzigingen naar een ERD om te zetten. Op deze manier heb ik ook het object model en het klasse diagram opgesteld. Ik ben gaan bedenken hoe ik de data zou structureren in een database. Ik heb de stap dus andersom genomen, van ERD naar klasse diagram. Daarbij heb dus niet gedaan wat ik bij mijn eerste lessen in object georiënteerd denken heb geleerd, namelijk uitgaan van de zelfstandige naamwoorden in de tekst van de probleemomschrijving voor het identificeren van de kandidaat klassen. Mijn gebrek aan ervaring op het gebied van UML heeft mij hierbij parten gespeeld.

11.2.3 Requirements document

In het Requirements document heb ik een doorwerk fout gemaakt vanuit het Business Architecture document. Ik heb het object model en het klasse diagram overgenomen, uitgebreid en aangepast.

Het object model heb ik uitgebreid met een actor welke ik ook bij de use cases heb toegevoegd, namelijk de beheerder. Het klasse diagram heb ik aangepast zoals ik het in een ERD gedaan zou hebben. Hier heb ik dus dezelfde denkfout gemaakt als bij het Business Architecture document.

In het Requirements document wordt de indruk gewekt dat het prototype van het controle formulier een resultaat is van de eerder opgestelde eisen. Dit is, echter, niet een geheel eerlijke representatie van de werkelijkheid. Reeds in de beginfase van

het project was er door de opdrachtgever een schets aan mij gegeven met daarin een beeld van hoe hij de applicatie voor zich zag.

11.2.4 Software Architecture document

Over het Software Architecture document ben ik redelijk tevreden. Ik denk dat ik een goede technische beschrijving van het systeem heb opgesteld. Ik denk alleen dat ik meer had kunnen schrijven over de wijze van implementatie van sommige procedures. Zoals ik al beschreef bij de evaluatie van de ontwikkeling in Delphi heb ik, bijvoorbeeld, tijdens het schrijven van de play-out controle module een figuur gemaakt welke de module modelleerde, deze had eigenlijk al in het Software Architecture document moeten staan.

11.2.5 Testplan

Over het testplan ben ik tevreden. Ik heb het helaas nog niet volledig uit kunnen voeren vanwege het feit dat de applicatie nog niet af is, maar nu ik het achteraf nog eens onder loep neem denk ik dat ik het in een volgende situatie weer zo op zou stellen.

11.2.6 Applicatie

Helaas is de applicatie nog niet volledig afgerond waardoor een evaluatie hiervan nog niet mogelijk is. Wel kan ik de afzonderlijke afgeronde onderdelen evalueren en daar kan ik niet anders dan tevreden over zijn. Ze doen tot dusver allemaal wat ze moeten doen. De XCP module is zelfs al, met succes, gebruikt in een ander project.

11.2.7 Handleiding

Zoals al eerder beschreven ben ik niet toegekomen aan het maken van een handleiding. Deze zal na de afstudeerperiode gemaakt en, samen met de applicatie, opgeleverd worden.

12 LEEREFFECTEN

Het uitvoeren van een afstudeeropdracht heeft een doel, namelijk het in de praktijk brengen van de opgedane kennis en het bewijzen dat na een aantal jaar studie een HBO werk- en denkniveau is behaald door de afstudeerder. Tevens doe je als afstudeerder nieuwe praktijk ervaring op met daaraan gekoppeld leereffecten. Deze leereffecten zijn besproken in dit hoofdstuk.

12.1 Rapportages

Rapporteren komt overal en altijd voor. Rapporten kunnen variëren van één A4tje tot de oplevering van een complete afstudeerscriptie. Als ik verslagen en rapporten van na mijn stage periode vergelijk met verslagen en rapporten zoals ik deze tijdens het afstuderen heb opgesteld, dan kan ik wel zeggen dat de kwaliteit, inhoud en structuur van deze verslagen en rapporten aanzienlijk verbeterd is. Dit komt mede door de hoeveelheid ervaring die ik heb opgedaan met Microsoft Word, maar ook vragen van collega's over, voor hen, onduidelijk beschreven zaken leerden mij beter formuleren. Verder heb ik in deze paragraaf beschreven welke leereffecten verbonden zijn aan de opgeleverde verslagen.

12.1.1 Plan van aanpak

Het grootste leereffect van een plan van aanpak is het leren plannen. Hiermee ben ik met bepaalde dingen de fout in gegaan. Bijvoorbeeld de geschatte omvang van de uiteindelijke realisatie in Delphi was groter dan ik vooraf had gepland. Weliswaar had ik gesteld dat de planning waarschijnlijk nog bijgesteld zou worden na verloop van tijd, maar in een ideale situatie wil je een plan van aanpak zo realistisch mogelijk opstellen. Tevens heb ik geleerd dat er voor studie zeker ook tijd vrij gemaakt moet worden. Zoals nu, werkend met een nieuwe methode en nieuwe technieken, is een bepaalde tijd vereist om je de nieuwe onderdelen eigen te maken.

12.1.2 Business Architecture document

Het Business Architecture document was het eerste verslag van redelijke omvang wat ik heb opgeleverd. De modellen die ik hierin heb opgenomen zijn gemaakt in Microsoft Visio, een pakket wat speciaal is ontwikkeld voor het construeren van modellen. Dit pakket heb ik gedurende het gehele ontwikkeltraject gebruikt om modellen mee te maken.

Daarnaast heeft het Business Architecture document, samen met het Requirements document en het Software Architecture document mij verplicht me te verdiepen in de RUP methode. Deze methode was voor mij volledig nieuw. In het verleden had ik weleens zijdelings te maken gehad met IAD, maar daarvoor was SDM de enige methode die ik beheerste.

Zoals al eerder beschreven, het object model en het klasse diagram wat opgenomen is in het Business Architecture document heb ik, achteraf, betiteld als niet goed.

Echter, hiervan heb ik wel geleerd hoe het maken van klasse diagrammen en object modellen wel aangepakt moet worden.

12.1.3 Requirements document

Het Requirements Document heeft mij het gebruik van UML geleerd. In het verleden heb ik de basis van UML wel geleerd, maar het gebruik hiervan is tijdens de studie tot een minimum beperkt. Helaas heb ik tijdens het opstellen van de diverse documenten niet goed gebruik gemaakt van de techniek. Daardoor heb ik wel geleerd mijn eigen werk kritisch te benaderen en heb ik inzicht gekregen in wat er fout is gegaan en hoe het wel had gemoeten.

12.1.4 Software Architecture document

Het Software Architecture document heeft mij geleerd de software van tevoren zo goed mogelijk te beschrijven zodat er tijdens de implementatie eigenlijk geen twijfel meer kan zijn over de wijze van implementatie.

12.1.5 Testplan

Het testplan heeft mij geleerd hoe het volledig testen van een systeem te benaderen. Dit had ik in het verleden nog nooit zo uitgebreid beschreven als bij dit project.

12.2 Het gebruik van RUP

Achteraf gezien heb ik het gebruik van RUP in eerste instantie als moeilijk ervaren, maar nu mijn afstudeerperiode zijn einde nadert heb ik wel het gevoel dat ik er enig vat op heb kunnen krijgen. Ik heb het gedurende het project niet altijd goed gebruikt, maar daar heb ik wel van geleerd hoe het wel moet.

12.3 Programmeren

12.3.1 Maken van de userinterfaces

Met Delphi is het eenvoudig om een nieuw scherm te maken en daar allemaal objecten op te plakken. De grote moeilijkheid is echter om deze objecten op een gestructureerde en voor de hand liggende manier te plaatsen. Ook moet het doel van een bepaald scherm eigenlijk in één oogopslag duidelijk zijn. Een scherm met als titel “Gegevens” met daarop een tekstvak en een knop met het opschrift “OK” zegt helemaal niets. Wanneer deze titel echter veranderd wordt in “Gegevens opslaan” en het formulier wordt uitgebreid met een stukje tekst wat toelicht dat er een bestandsnaam ingevuld dient te worden in het tekstvak, dan wordt het allemaal een stuk duidelijker voor een gebruiker. Zo heb ik in overleg met mijn stagebegeleider en een grafisch ontwerper de meeste schermen ontworpen. Het praten en brainstormen over dit onderwerp heeft mij wel veel geleerd over hoe een scherm het beste opgebouwd kan worden en wat het meest overzichtelijk is voor een gebruiker. Tevens heb ik geleerd om regelmatig om advies te vragen aan collega’s. Zij kunnen namelijk onbevooroordeeld kijken naar een scherm. Als zij dan niet vrij snel kunnen zien waar het scherm voor is, dan weet ik dat ik iets niet goed gedaan heb en dan vraag ik om advies.

12.3.2 Maken van de database

Ik had al wat ervaring met het ontwerpen van databases in Access 2000, echter, bij dit project had ik niet de beschikking over Access, maar over MySQL. Dit DBMS verplichte mij weer eens met de hand create table-queries op te stellen. Iets wat ik al lange tijd niet meer had gedaan. Hiervoor was enige onderzoek wel vereist.

12.3.3 Communicatie met de units

Het maken van de XCP communicatie module heeft mij heel veel geleerd over het implementeren van protocollen. Hoewel in Delphi standaard een UDP communicatie module zit moest er nog zeer veel gebeuren voordat er een XCP communicatie module ontstond. De summiere beschrijving van de leverancier van de units heeft mij geleerd hoe ontzettend belangrijk goede documentatie is wanneer er met meerdere mensen met één onderdeel gewerkt wordt. Tevens heeft de redelijk beperkte documentatie mij geleerd creatief dit soort problemen te benaderen en deze gestructureerd aan te pakken

12.4 Nevenactiviteiten

12.4.1 Ontwikkelen van de OSD-applicatie

Gedurende mijn afstudeerperiode is mij gevraagd een kleine applicatie te schrijven die door een gebruiker ingevoerde teksten zou weg schrijven als OSD-bestanden (Bitmap bestanden die op een unit getoond kunnen worden) en deze vervolgens te tonen op een unit. Hierbij heb ik geleerd hoe praktisch het kan zijn om herbruikbare stukken programmatuur te schrijven. Ik kon namelijk direct gebruik maken van de door mij ontwikkelde XCP DLL. Speciaal voor deze applicatie heb ik nog een DLL geschreven, namelijk één die van tekst Windows bitmap bestanden kan maken (BMP) en die van BMP bestanden Adtec On-Screen Display bestanden (OSD) kan maken. Dit heb ik gedaan aan de hand van betere documentatie van de leverancier van de units.

Echter ik heb hiervan ook geleerd dat, ook relatief kleine, projecten niet onderschat mogen worden. Dit project heeft toch een kleine anderhalve week in beslag genomen terwijl ik dacht dat ik het in een halve week klaar zou hebben.

LITERATUURLIJST

Titel	Auteur	Uitgever
Soloist-2 User Manual	onbekend	Adtec Inc.
Database Systemen Voor De Praktijk	Prof. Dr. J.A. Vandenbulcke	Kluwer Bedrijfswetenschappen
Fundamentals Of Software Engineering	C. Ghezzi	Prentice Hall
Delphi In A Nutshell	R. Lischner	O'Reilly and Associates
Rational Unified Process Made Easy	P. Kroll, P. Kruchten	Addison Wesley

BIJLAGES

Ontwikkelen Ad-2-Add Play-out Monitor

Auteur:
Dhr. M.J.M. van Meurs (99008956)
Haagse Hogeschool

Opdrachtgever:
Dhr. F. Ouwendijk
Media Choice

Bedrijfsmentor:
Dhr. F. Voskamp
Media Choice

Examinatoren:
Dhr. A. van der Molen
Dhr A.G.J. Vinkesteijn
Haagse Hogeschool

Datum:
8 oktober 2004

BIJLAGE A.
PLAN VAN AANPAK

INHOUDSOPGAVE

1 INLEIDING

In dit document zal ik de activiteiten met de daarop volgende producten vastleggen, die ik tijdens dit project uit zal voeren. Tevens staat in dit document beschreven wat de achtergrond- en de uiteindelijke doelstelling van het project is.

2 DOELSTELLING VAN DE OPDRACHT

Media Choice zou graag de beschikking krijgen over een applicatie die periodiek, meerdere malen per uur, controleert of alle remote units de juiste taken uitvoeren en/of uitgevoerd hebben. Dit dient te gebeuren door het ophalen, uit de remote units, van variabelen en logfiles. De opgehaalde data dient daarna verwerkt en opgeslagen te worden in een database. De gegevens in de database worden vergeleken met elkaar (correlatie) en vooraf gedefinieerde waardes. Met de resultaten van deze analyses dienen regelmatig rapporten gegenereerd te worden. Tevens dient er, wanneer er een 'ernstige' storing wordt geconstateerd, een technische medewerker gealarmeerd te worden door middel van e-mail en/of sms.

3 OP TE LEVEREN PRODUCTEN

Gedurende de loop van het project zullen we de volgende producten opleveren:

- Business Architecture Document;
- Requirements Document;
- Software Architecture Document;
- Test Plan Document;
- “Ad-2-Add Play-out Monitor”-applicatie;
- Beknopte handleiding.

4 TE GEBRUIKEN METHODES EN TECHNIEKEN

Tijdens de ontwikkeling zal, als methode, gebruik gemaakt worden van Rational Unified Process (RUP). Als techniek zal gebruik gemaakt worden van onderdelen uit de Unified Modeling Language (UML) techniek.

Om de opdracht uit te voeren zal er gebruik gemaakt worden van Rational Rose, voor het maken en onderhouden van UML diagrammen en de Delphi-source, en Borland Delphi 7, voor de ontwikkeling van de applicatie in Delphi.

De opdracht zal uitgevoerd worden volgens de RUP methode in een object georiënteerde omgeving.

5 STANDAARDS, RICHTLIJNEN EN PROCEDURES

De lay-out van de documenten zal er al volgt uit komen te zien: op het voorblad staat een titel en subtitel, de naam van de auteur met diens studentnummer, de

bedrijfsmentor, de examinatorren, de datum en een versienummer vermeld. Alle documenten zullen in het Nederlands opgesteld worden.

De technische documentatie van de te ontwikkelen applicatie zal zowel binnen als buiten de programmacode gebeuren. Binnen de code zal dit gebeuren door zogenaamde ‘comments’ die de regels code vergezellen. Dit zal gebruikt worden om de werking van de programmacode uit te leggen en toe te lichten. Buiten de code zullen de opgeleverde schema’s en beschrijvingen die de gebruikte ontwerpmethode voorschrijft de technische documentatie vormen.

6 RISICO’S

Gezien het feit dat het project door slechts één persoon wordt uitgevoerd bestaat de kans dat bij het uitvallen van deze persoon, bijvoorbeeld door ziekte, het gehele project niet of niet tijdig afgerond kan worden.

7 PLANNING

De volgende planning zal gehanteerd worden:

- Opstellen van een Plan van Aanpak 07 jun – 09 jun;
- Het interviewen van diverse medewerkers 10 jun – 11 jun;
- Verkennen mogelijkheden m.b.t. het uitlezen van de units 14 jun – 18 jun;
- Maken van een Business Architecture Document 21 jun – 02 jul;
- Maken van een Requirements Document 05 jul – 16 jul;
- Maken van een Software Architecture Document 19 jul – 30 jul;
- Maken van een Test Plan Document 02 aug – 04 aug;
- Programmeren 05 aug – 15 sep;
- Testen 16 sep – 17 sep;
- Schrijven van een beknopte handleiding 20 sep – 24 sep;
- Afronden van het eindverslag 27 sep – 08 okt.

8 BETROKKENHEID VAN OPDRACHTGEVER, BEDRIJFSMENTOR EN PROJECTUITVOERDER

Van de projectuitvoerder wordt verwacht dat hij met ideeën, oplossingen, programmeerwerk en documentenbeheer dit project tot een goed einde zal brengen. De opdrachtgever en de bedrijfsmentor spelen hierin een grote rol vanwege hun kennis van, respectievelijk: de opdracht en de apparatuur waarmee gewerkt zal gaan worden. Daarnaast zal de bedrijfsmentor ook een rol gaan spelen in dit geheel in verband met de begeleiding van de uitvoerder in het proces. Van zowel de opdrachtgever als de bedrijfsmentor wordt verwacht dat zij regelmatig, minimaal eenmaal per week, beschikbaar zijn voor de uitvoerder.

9 PROJECTORGANISATIE

Binnen projectgroepen is er in veel gevallen sprake van een zekere taakverdeling. Echter, gezien het feit dat dit project door slechts één persoon uitgevoerd wordt is er in dit geval geen sprake van een duidelijke projectorganisatie.

10 KWALITEIT

Het testen van de kwaliteit van de te schrijven applicatie zal doormiddel van zowel black- als whitebox testen plaatsvinden. Hierdoor kan worden bepaald of het programma aan alle gestelde eisen voldoet.

Bij ‘blackbox’ testen, of functioneel testen, wordt de input van een specifieke functie vergeleken met de output ervan. Deze output moet conform de vooraf vastgelegde specificaties zijn. Er wordt niet gekeken hoe de functie zich intern gedraagt.

Bij ‘whitebox’ testen, of structureel testen, echter, wordt gecontroleerd of programmacode voldoet aan de eisen met betrekking tot syntactische en grammaticale regels die door de gekozen programmeertaal opgelegd worden.

Tests die hiernaast worden gebruikt om de kwaliteit van de applicatie te waarborgen zijn:

- Testen tijdens de bouw. Tijdens het ontwikkelen wordt regelmatig gecontroleerd of de applicatie, voor zover het af is, doet wat het moet doen;
- Testen van de afzonderlijke functies. Als een deel functie of procedure van de applicatie afgerond is, test de ontwikkelaar deze uitvoerig;
- Gebruikerstest. Bij een gebruikerstest wordt de toekomstige gebruikers gevraagd in een testsituatie realistische gegevens in te voeren in het systeem en commentaar te leveren;

Er zal door de projectuitvoerder een testplan worden opgesteld waarin deze verschillende vormen van testen worden opgenomen. Aan de hand van deze testen wordt de kwaliteit van de applicatie geverifieerd.

11 OVERDRACHT EN AFSLUITING

Het project wordt afgerond op 8 oktober 2004. De opdrachtgever zal dan, voor zover beschikbaar en afgerond, de applicatie broncode, de applicatie in binaire vorm, de technische documentatie en een beknopte gebruikershandleiding ontvangen. De examinatorren en de bedrijfsmentor zullen dan het eindverslag ontvangen.

BIJLAGE B.
BUSINESS ARCHITECTURE

INHOUDSOPGAVE

1 INLEIDING

Dit is het Business Architecture document voor de Ad-2-Add Play-out Monitor. Hoewel het business architecture document technisch niet beschouwd wordt als een onderdeel van de documentatie van APM stelt het wel de achtergrond en de context voor de applicatie vast. Het business architecture document dient als het model van de organisatie waar APM ingezet zal worden.

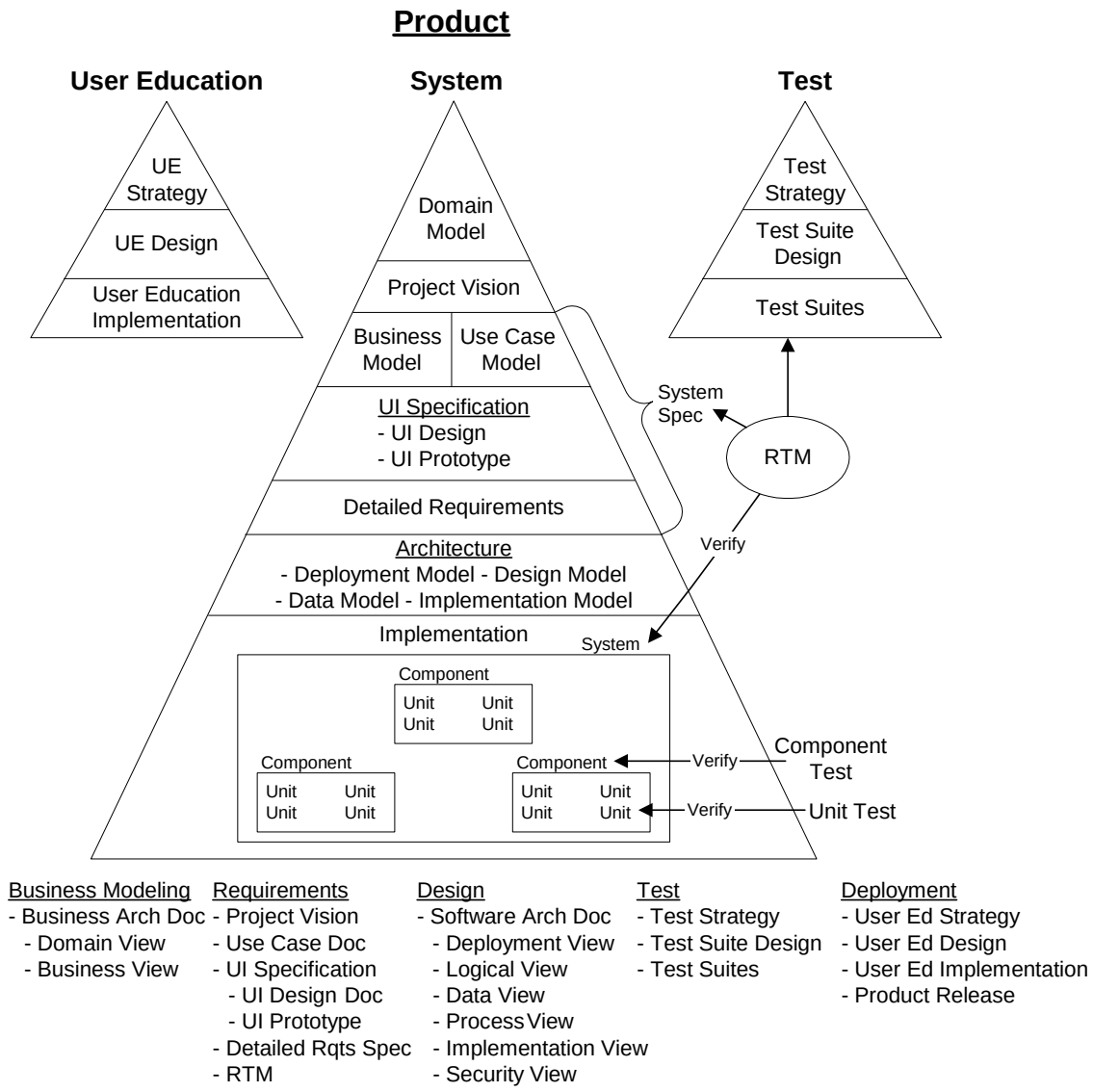
1.1 Doel

Dit document voorziet in een uitgebreid overzicht in de architectuur van het systeem, gebruik makend van een aantal verschillende views op de architectuur om zodoende inzicht te krijgen in verschillende aspecten van het systeem. Het is bedoeld om belangrijke beslissingen betreffende de architectuur in beeld te krijgen.

1.2 Bereik

Dit document geeft inzicht in hoe het systeem in zijn omgeving staat en welke beslissingen op basis daarvan worden genomen.

In het volgende model is aangegeven wat de RUP producten uit de verschillende documenten zijn en hoe de relatie tussen deze producten onderling is.



Figuur 1-1 RUP producten en hun relaties

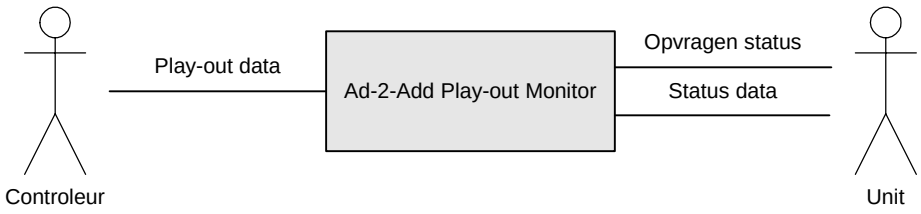
Het business architecture document bestaat uit het domein model, het model van de context waarin het systeem moet gaan functioneren

2 DOMEIN MODEL

Het domein model beschrijft in het kort, op een object georiënteerde wijze, de bedrijfsomgeving waarin APM zal gaan functioneren. Een domein model beschrijft wat het bedrijfsprobleem is, terwijl een business proces model beschrijft hoe dit probleem opgelost kan worden. Hierdoor bevat een domein model geen details zoals gegevens over personeelsbezettingen, back office systemen of mechanismen voor interactie. Het domein model, echter, definieert de essentiële entiteiten en de interactie tussen deze activiteiten en de organisatie. De wijze waarop communicatie over en weer plaats vindt wordt beschreven in de business en use case modellen.

2.1 APM Domein Model Context Diagram

In het context diagram worden de actors bepaald en hun relatie met het systeem gedefinieerd.



Figuur 2-1 APM context diagram

2.1.1 Domein actors

In de volgende tabel staat een overzicht van de actoren die te maken hebben met APM inclusief een beschrijving. Een actor is een entiteit die buiten het systeem staat en die direct communiceert met het systeem. Een actor kan zowel een mens, als ook een ander systeem zijn. Één mens of systeem kan ook meerdere actoren representeren. Vanuit het systeem gezien zijn de actoren de representatie van de complete buitenwereld. Het systeem communiceert alleen met actoren.

Actor	Rol
Controleur	Medewerker(s) die de rapportage van het systeem analyseert en interpreteert.
Unit	Het video uitspeel systeem dat op verzoek data betreffende de huidige status van het systeem verstrekt.

2.2 Domein Use Cases

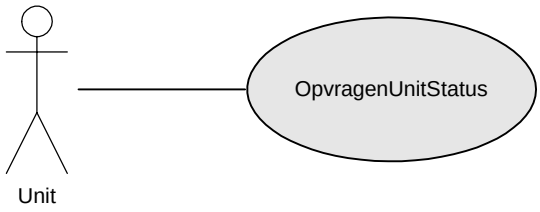
In de volgende tabel staan de use cases zoals die binnen APM worden gedefinieerd. Use cases binnen UML zijn een middel om de functionele eisen die gesteld worden aan het systeem weer te geven. De nu volgende use cases beschrijven hoe de gebruiker met het systeem om zal gaan.

Een use case is een beschrijving van een bepaalde wijze waarop het systeem gebruikt kan worden. Een use case wordt beschreven in natuurlijke taal en is daarom

informeel. In het vervolg van de systeemontwikkeling worden de use cases als uitgangspunt genomen voor de formelere sequence diagrammen. Tenslotte kunnen de use cases worden gebruikt als testcases voor het uiteindelijk opgeleverde systeem. Iedere beschreven use case dient met behulp van het systeem uitgevoerd te kunnen worden. De verzameling use cases dient dan ook compleet te zijn; iedere gewenste systeemfunctie dient te worden beschreven.

Use Case ID	**Titel**
0001	Opvragen unit status
0002	Invoeren unit status data
0003	Verstrekken play-out data
Use Case ID	**Titel**
0001	Opvragen unit status
0002	Invoeren unit status data
0003	Verstrekken play-out data

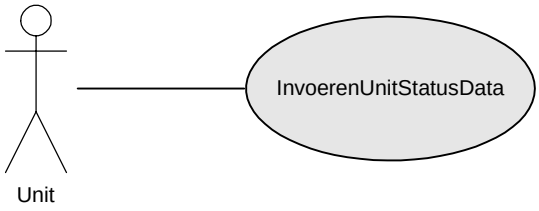
2.2.1 Use Case 0001 – Opvragen unit status



Figuur 2-2 Use case OpvragenUnitStatus

1. Er is minimaal één unit ingevoerd.
2. Het systeem doet een verzoek voor de unit status gegevens bij een unit.

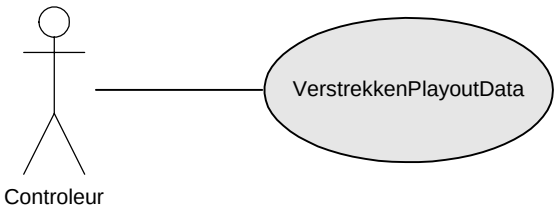
2.2.2 Use Case 0002 – Invoeren unit status data



Figuur 2-3 Use case InvoerenUnitStatusData

1. De unit heft een verzoek ontvangen van het systeem om status gegevens te verstrekken.
2. De unit voert de gevraagde status data in het systeem in.
3. De unit status data wordt opgeslagen in een database.

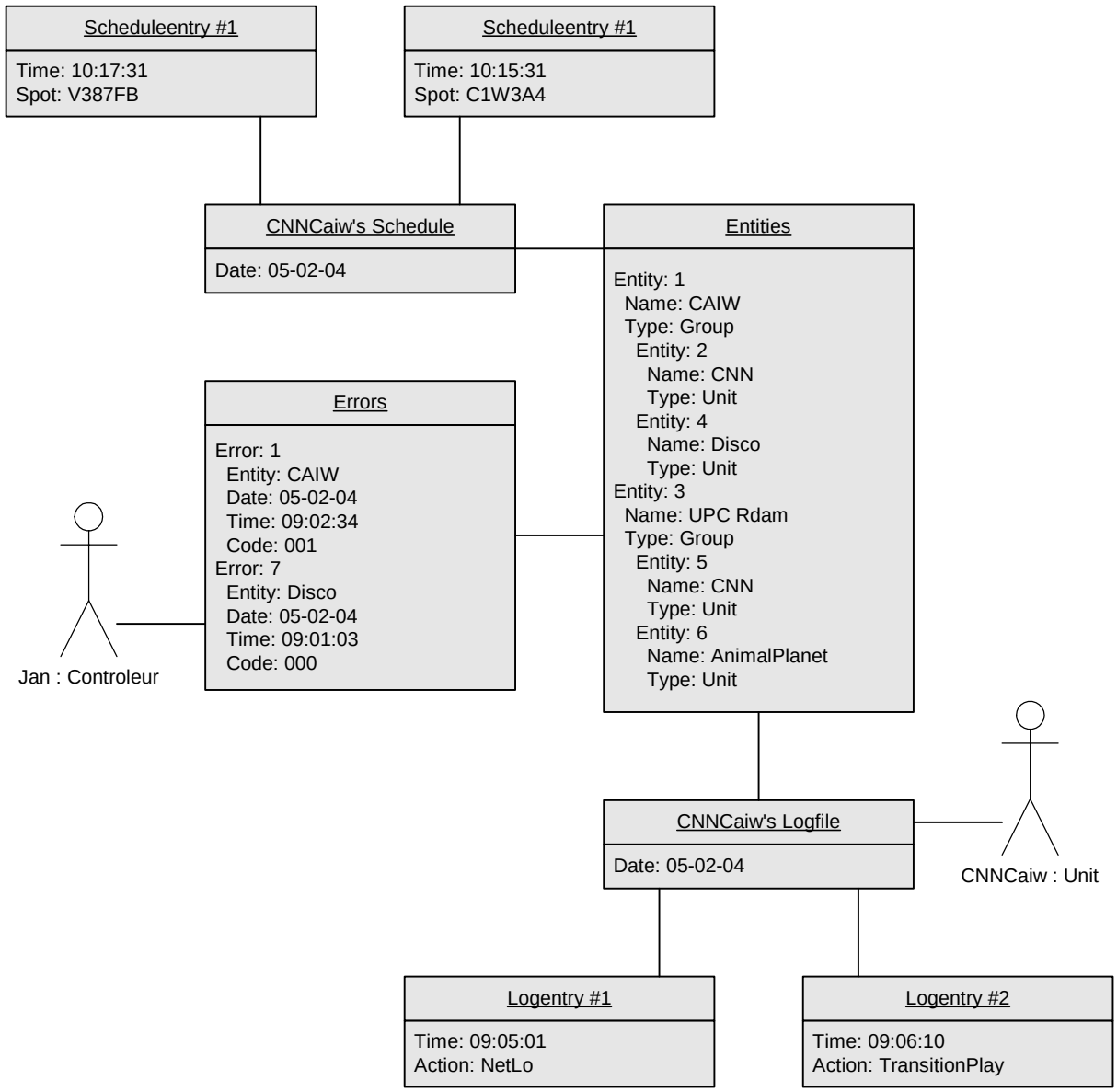
2.2.3 Use Case 0003 – Verstrekken play-out data



Figuur 2-4 Use case VerstrekkenPlayoutData

1. Het systeem heeft van minimal één unit status data ontvangen.
2. Het systeem analyseert de ontvangen data en trekt hieruit conclusies.
3. De controleur krijgt van het systeem de conclusies en brongegevens gepresenteerd.

2.3 Domein Object Model Voorbeeld

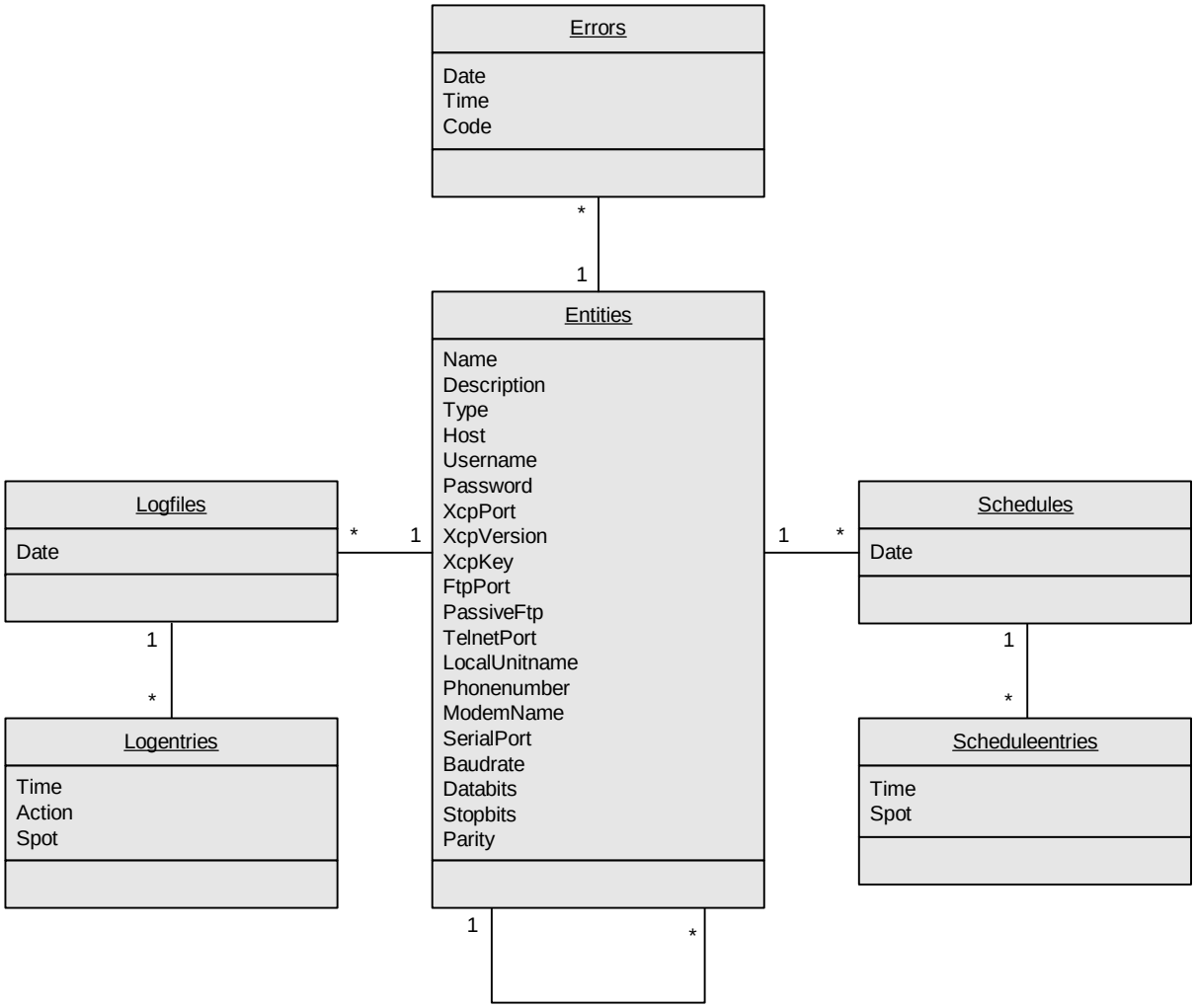


Figuur 2-5 Domein object model voorbeeld

CNNCaiw is een unit die op verzoek van het systeem zijn ‘logfiles’ er in invoert. Deze logfiles bestaan uit diverse ‘logentries’ waarin gebeurtenissen in de unit op een bepaald moment beschreven staan. De logfiles staan gekoppeld aan de ‘entities’. In de entities wordt beschreven welke units aan het systeem gekoppeld zijn en hoe. In dit object worden de in de logfile beschreven logentries vergeleken met de ‘schedules’ en ‘scheduleentries’ en bepaald of deze geen onverwachte informatie bevatten en/of de verbindingen naar de units nog intact zijn. Indien dit niet het geval is wordt in ‘errors’ een nieuwe regel toegevoegd waarin beschreven wordt welke error bij welke entiteit hoort en wat de error inhoudt. Jan is controleur en heeft een continu zicht op de errors en kan ingrijpen wanneer een error optreed.

2.4 Domein Object Model (Class Diagram)

Alle instanties en relaties uit het hierboven gemodelleerde object model zijn geëxtrapoleerd naar het onderstaande class diagram.



Figuur 2-6 Klasse diagram

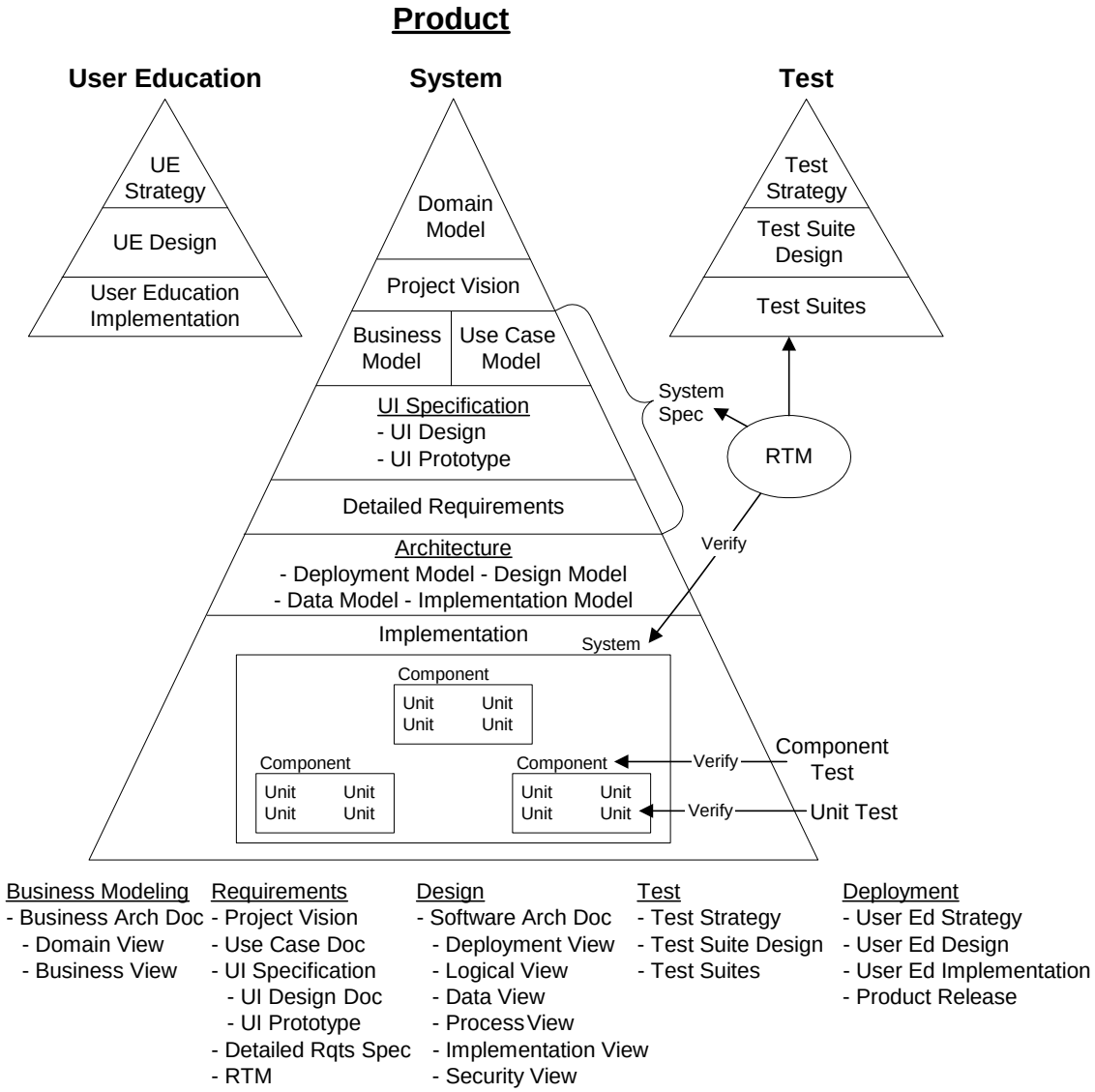
Als basis voor dit class diagram geldt ‘entities’. Deze class bevat alle gegevens die benodigd zijn om verbinding te maken met een unit of een group. Entities heeft een relatie met zichzelf, omdat, wanneer een object van de class entities van het type ‘group’ is, deze een relatie kan hebben met een ander object van de class entities. Er kan bijvoorbeeld een group gevormd worden met de naam ‘Caiw’. Deze group kan objecten van de class entities van het type ‘unit’ onder zich hebben. Op deze manier kunnen eigenschappen van de group overerft worden door één of meerdere units. Naast de relatie met zichzelf bevat entities ook een relatie met ‘logfiles’ en ‘schedules’. Deze relatie geldt alleen wanneer het object van de class entities van het type unit is. De relatie met ‘errors’ geldt voor alle objecten van de class entities.

BIJLAGE C.
REQUIREMENTS

INHOUDSOPGAVE

1 INLEIDING

Dit is het requirements document voor APM. Het requirements document volgt na het business architecture document. De onderdelen in de systeem driehoek, in het diagram hieronder, van ‘Project Vision’ t/m ‘Detailed Requirements’ worden in dit document beschreven. Er zal worden beschreven wat de achtergrond van het project is, hoe de gewenste situatie eruit ziet, hoe de userinterfaces eruit gaan zien en wat de gedetailleerde eisen aan het systeem zijn.



Business Modeling	Requirements	Design	Test	Deployment
- Business Arch Doc	- Project Vision	- Software Arch Doc	- Test Strategy	- User Ed Strategy
- Domain View	- Use Case Doc	- Deployment View	- Test Suite Design	- User Ed Design
- Business View	- UI Specification	- Logical View	- Test Suites	- User Ed Implementation
	- UI Design Doc	- Data View		- Product Release
	- UI Prototype	- Process View		
	- Detailed Rqts Spec	- Implementation View		
	- RTM	- Security View		

Figuur 1-1 RUP producten en hun relaties

In bovenstaand figuur staat beschreven (top-down) uit welke documenten RUP bestaat en wat de volgorde en samenhang tussen deze documenten is.

Project Visie: Hierin worden de achtergronden van het project beschreven. Onder andere wordt beschreven wat het uiteindelijke doel is van het project en wat de succescriteria van het project zijn.

Use Cases**:** Modelleren de gewenste situatie. Hoe gaat de interactie tussen de gebruikers en het systeem eruit zien wanneer het systeem in geïmplementeerd.

User Interface Specificatie**:** De specificatie van de userinterfaces opgedeeld in twee stukken:

- Ontwerp van de userinterface;
- Prototype van de userinterface.

Gedetailleerde eisen**:** Specificeert op het laagste niveau de eisen aan het systeem.

Hoewel de Project Visie de doelen en financiële onderbouwing van het project bevat, worden deze niet gezien als formele systeemeisen. Eerder vormt het een basis voor de systeemeisen, het ontwerp, de implementatie en de testen. Het use case model, de userinterface specificatie en de documenten met de gedetailleerde eisen bevatten de formele systeemeisen. Als een resultaat hiervan vormen deze documenten tezamen de Systeem Specificaties.

Tevens bevat dit Requirements Document een Requirements Traceability Matrix (RTM) die zowel systeem eisen als systeem testen in kaart brengt om ervoor te zorgen dat alles is beschreven.

2 PROJECT VISIE

Het doel van de projectvisie is het definiëren van de uiteindelijke missie van het project vanuit het perspectief van het bedrijf. Al het werk dat wordt gedaan aan het project, direct of indirect, ondersteunt de doelen die zijn aangegeven in de projectvisie. De projectvisie bestaat uit vier onderdelen:

- Bedrijfsachtergrond en overzicht;
- Probleemomschrijving;
- Project missie;
- Project doelen, inclusief succes criteria.

Het definiëren van een duidelijke, haalbare missie voor het project is essentieel. Wanneer op een juiste manier wordt gecommuniceerd, is de projectvisie een goede leidraad voor het maken van beslissingen op het gebeid van planning, mogelijkheden en kosten. Het wordt gedurende de gehele cyclus van ontwikkeling gebruikt om problemen op te lossen betreffende gedetailleerde eisen, ontwerp, implementatie en testen; degenen die de visie ondersteunen worden in het document opgenomen, anderen worden weggelaten of veranderd.

De projectvisie omvat de uiteindelijke motivatie om in het project te investeren. Het definieert doelen van het project in termen van het bedrijf, die gebruikt worden om beslissingen te nemen tijdens de gehele ontwikkelingscyclus. Als er wordt onderzocht welke opties en toepassingen het systeem moet hebben moet de projectvisie in gedachten worden gehouden.

2.1 Bedrijfsachtergrond en overzicht

Media Choice is een 6 jaar geleden opgericht bedrijf dat door geheel Nederland ‘ad-insertion’ verzorgt. Ad-insertion is het invoegen van lokale/regionale reclameblokken op internationale televisiezenders. Media Choice heeft circa 10 medewerkers.

2.2 Probleemomschrijving

Media Choice maakt gebruik van de hardwarematige MPEG-video decoders van Adtec Inc. (ook wel ‘units’ genaamd) om de geprogrammeerde reclame blokken uit te zenden. Deze units zijn door Nederland, België en het Verenigd Koninkrijk verdeelt bij de diverse ‘head-ends’ van kabelmaatschappijen. Head-ends zijn de locaties van kabelmaatschappijen waar vandaan het televisiesignaal verstuurd wordt naar de kabelabonnees. De units zijn op afstand te beheren via Internet of via de telefoonlijn (m.b.v. een modem). Omdat de organisatie momenteel ongeveer 50 units ingezet heeft is het praktisch onmogelijk om met de hand te controleren of alle geprogrammeerde reclame blokken ook daadwerkelijk hebben gespeeld en of de juiste spots zijn uitgezonden. Hiervoor zou er namelijk op iedere unit ingelogd moeten worden om een aantal variabelen en logfiles op te halen om deze te analyseren en te vergelijken.

2.3 Project Missie

De project missie voor APM is het inzicht verschaffen aan Media Choice omtrent de status van de ingezette apparatuur. Dit zodat bij eventuele problemen zo snel mogelijk in gegrepen kan worden.

2.4 Project Doelen

Het doel van de afstudeeropdracht is het ontwikkelen van een applicatie die periodiek, meerdere malen per uur, controleert en analyseert of alle remote units de juiste taken uitvoeren en/of uitgevoerd hebben en de resultaten hiervan grafisch presenteert. Dit dient te gebeuren door het ophalen, uit de remote units, van variabelen en logfiles. De opgehaalde data dient daarna verwerkt en opgeslagen te worden in een database. De gegevens in de database worden vergeleken met elkaar (correlatie) en vooraf gedefinieerde waardes. Met de resultaten van deze analyses dienen regelmatig rapporten gegenereerd te worden. Tevens dient er, wanneer er een storing wordt geconstateerd (n of meerder units hebben niet of onvolledig de vooraf gedefinieerde spots gespeeld), een technische medewerker gealarmeerd te worden.

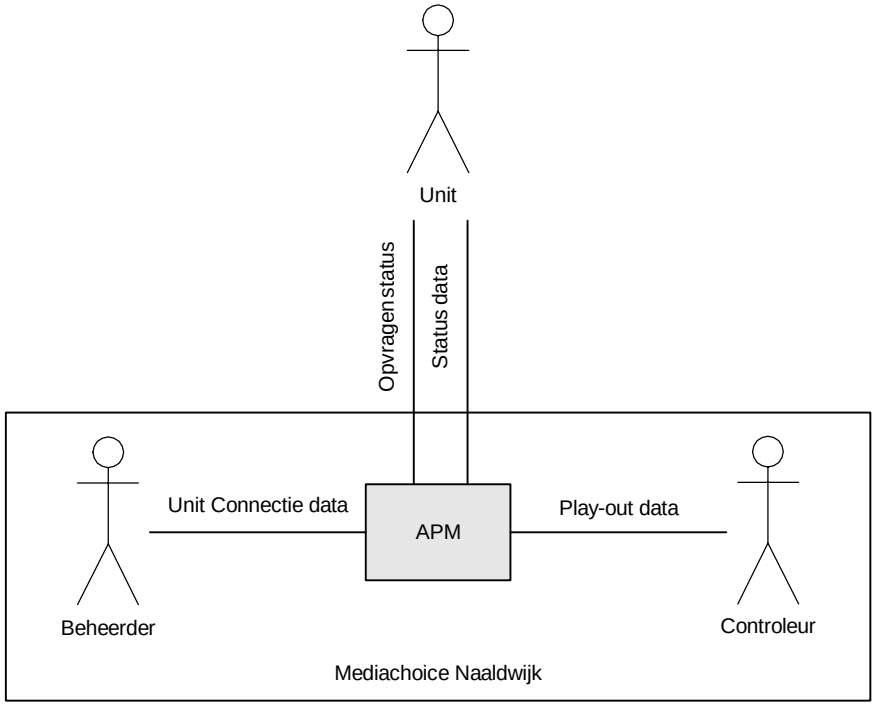
3 TOEKOMSTIGE USE CASES

Het use case model beschrijft het hoogste niveau van detail van de systeemeisen doormiddel van het modelleren van de interne bedrijfsprocessen uitgaande van een werkend systeem in de toekomstige situatie. Het beschrijft de interne systemen en de verschillende gebruikersrollen die deel uitmaken van de processen binnen het systeem. Het richt zich voornamelijk op de interactie tussen APM en de actoren (een entiteit die buiten het systeem staat en die direct communiceert met het systeem) binnen het systeem.

Het use case model dient de volgende doelen:

- Het geeft het systeem op het hoogste niveau weer inclusief relaties met andere systemen;
- Het beschrijft de actoren die deel uitmaken van het systeem, zowel extern als intern;
- Het beschrijft hoe het systeem bepaalde problemen bij de bedrijfsprocessen oplost doormiddel van use cases;
- Het geeft een aanzet voor het ontwerp van de gebruikersinterface;
- Het voorziet in een framework op hoog niveau voor het specificeren van gedetailleerde eisen in de Detailed Requirements.

3.1 APM Toekomstige Situatie Use Case Context Diagram

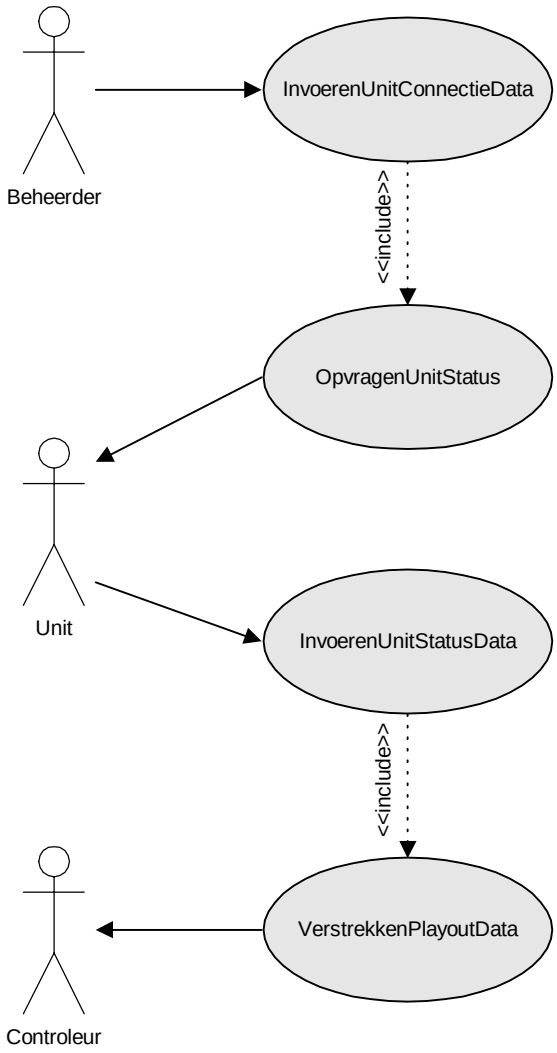


Figuur 3-1 Toekomstige situatie use case context diagram

De beheerder logt in op APM om unit connectie gegevens waarmee het systeem zal moeten gaan werken in te voeren. De controleur logt in om in de gaten te houden of

de status data die van de units, na aanvraag, ontvangen wordt de data is die verwacht wordt.

3.2 Gewenste Use Case



Figuur 3-2 Gewenste situatie use case

Use Case ID	Titel
0001	Invoeren unit connectie data
0002	Opvragen unit status
0003	Invoeren unit status data
0004	Verstrekken play-out data

3.2.1 Use case 0001 – Invoeren unit connectie data

1. De beheerder voert de gegevens benodigd om een verbinding met de unit te kunnen maken in.

3.2.2 Use case 0002 – Opvragen unit status

1. Er is minimaal één unit ingevoerd.
2. Het systeem doet een verzoek voor de unit status gegevens bij een unit.

3.2.3 Use case 0003 – Invoeren unit status data

1. De unit heft een verzoek ontvangen van het systeem om status gegevens te verstrekken.
2. De unit voert de gevraagde status data in het systeem in.
3. De unit status data wordt opgeslagen in een database.

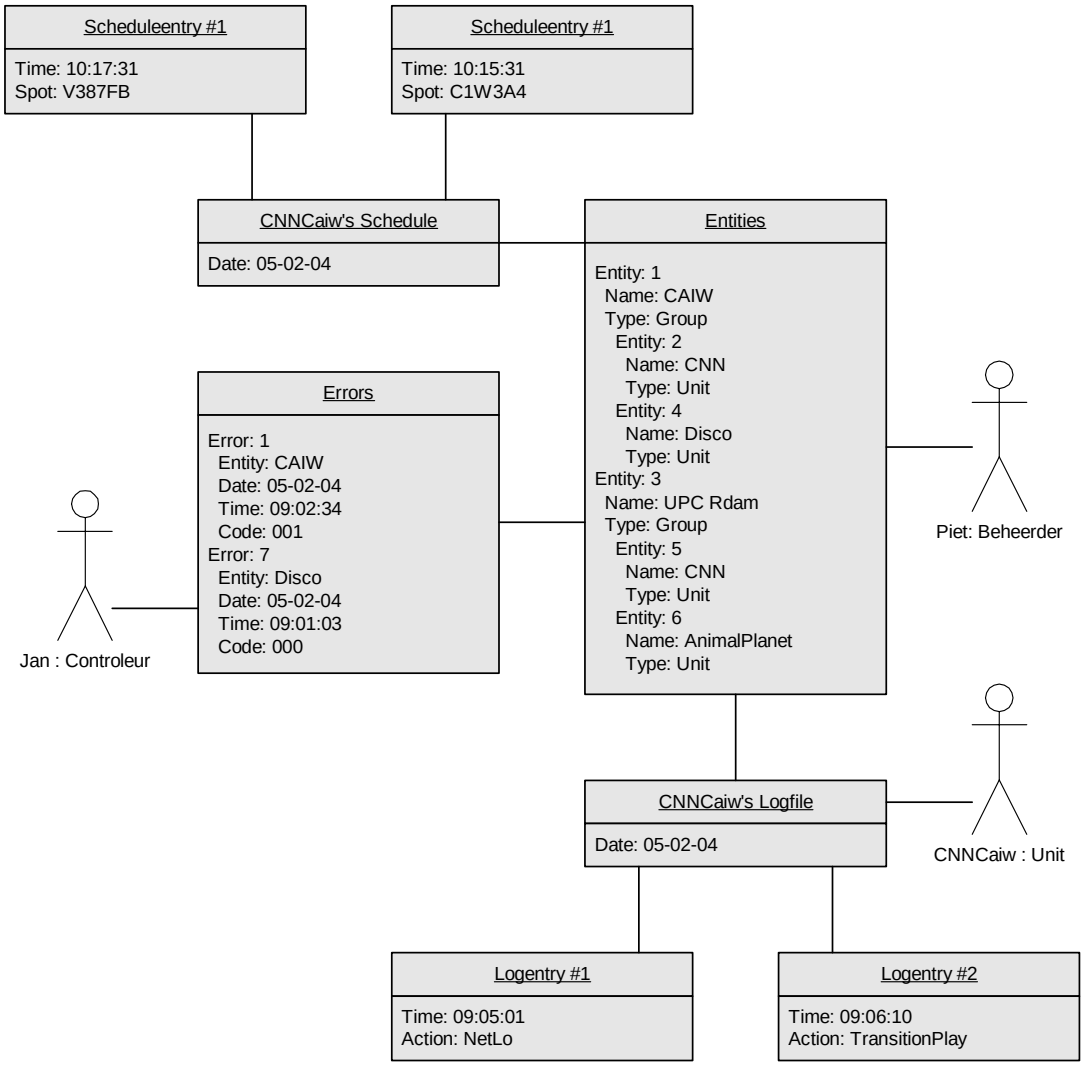
3.2.4 Use case 0004 – Verstrekken play-out data

1. Het systeem heeft van minimal één unit status data ontvangen.
2. Het systeem analyseert de ontvangen data en trekt hieruit conclusies.
3. De controleur krijgt van het systeem de conclusies en brongegevens gepresenteerd.

3.3 Systeem Interfaces

Behalve de communicatie met de units (die zijn opgenomen in het systeem) heeft het systeem geen interfaces met andere systemen.

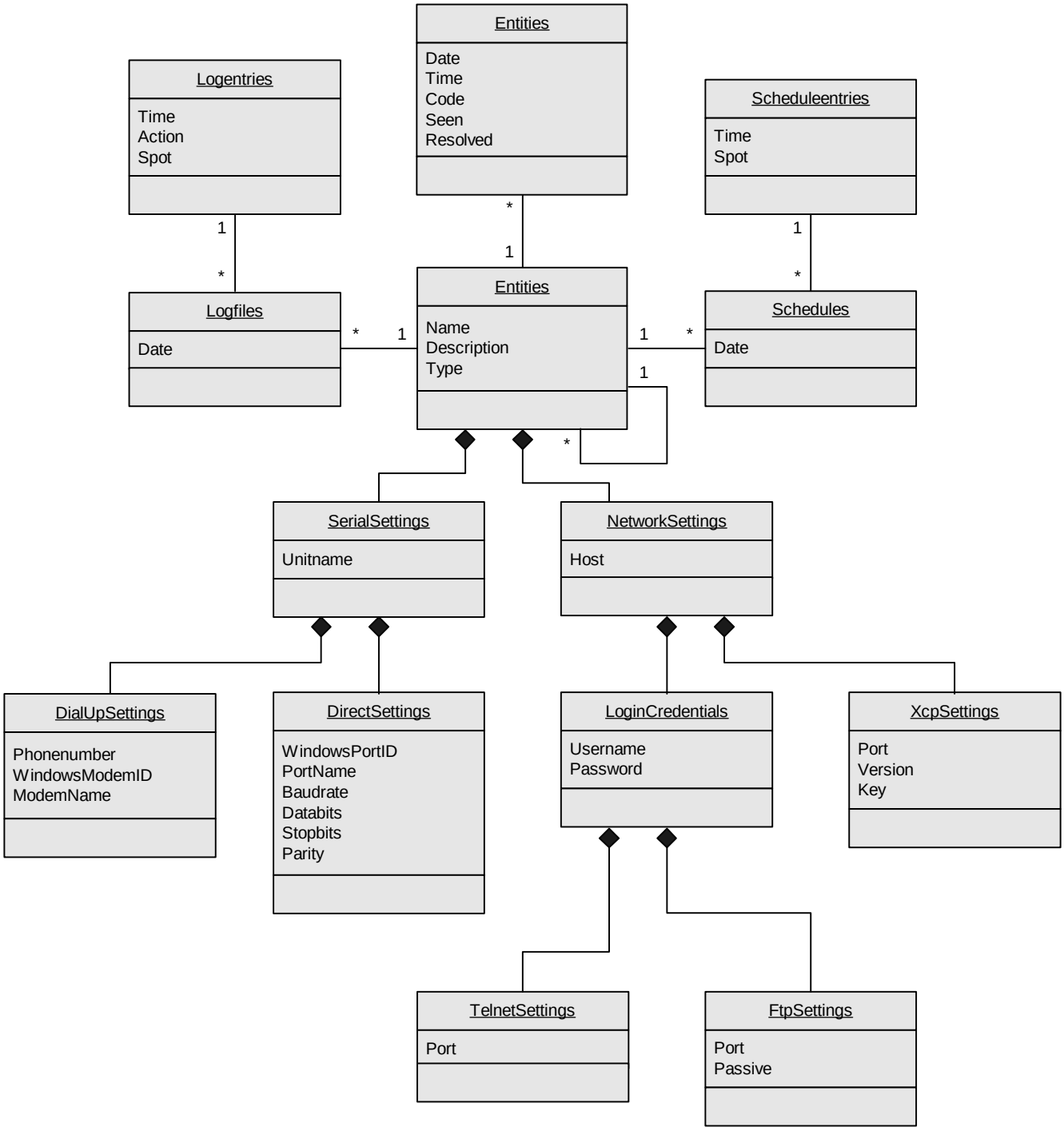
3.4 Use Case Object Model



Figuur 3-3 Use case object model

In bovenstaande figuur is te zien hoe de drie actorren ieder informatie in het systeem brengen of uit het systeem ophalen. Beheerder Piet is verantwoordelijk voor het bijhouden van alle units en unit groepen die bestaan. Unit CNNCaiw is een unit zoals beschreven in ‘Entities’. Deze voert, op verzoek van het systeem, regelmatig alle log informatie in het systeem. Controleur Jan leest, naar aanleiding van een waarschuwing vanuit het systeem, welke fout er opgetreden is. De controleur zal, maar dit valt buiten beschouwing van dit systeem, de foutmelding interpreteren en eventuele actie ondernemen.

3.5 Use Case Model Class Diagram



Figuur 3-4 Use case model class diagram

In bovenstaande figuur is het object diagram uit de vorige paragraaf uitgewerkt in een class diagram. Hierin is duidelijk te zien hoe entities is opgesplitst in compositie relaties. Dit, omdat niet voor iedere unit dezelfde variabelen ingesteld zullen worden en, aangezien overerving van andere entities een eis is, er sprake zou kunnen zijn van veel overhead.

4 USERINTERFACE SPECIFICATIES

In dit hoofdstuk worden de specificaties van de userinterface (UI) beschreven. Dit wordt in twee gedeelten gedaan:

- UI ontwerp; hierin worden de conceptuele eisen en wensen voor de userinterface beschreven, inclusief userinterface standaarden;
- UI prototype; een uitvoerbare specificatie die het mogelijk maakt deze te testen zodra het is geïmplementeerd.

4.1 Ontwerp

Door gebruik te maken van de actoren en de takenanalyse zoals die is beschreven in het use case model, kan in het UI ontwerp een ontwerp worden gemaakt van de achtergrond en omgeving waarin de eindgebruikers deze taken uit zullen gaan voeren. Uitgaande van dit ontwerp wordt vervolgens het prototype van de userinterface gemaakt. In het UI ontwerp wordt beschreven hoe de UI er voor de gebruiker uit zal gaan zien en hoe de structuur en werking van de UI wordt. In de Detailed Requirements Specification worden de details van de UI eisen beschreven.

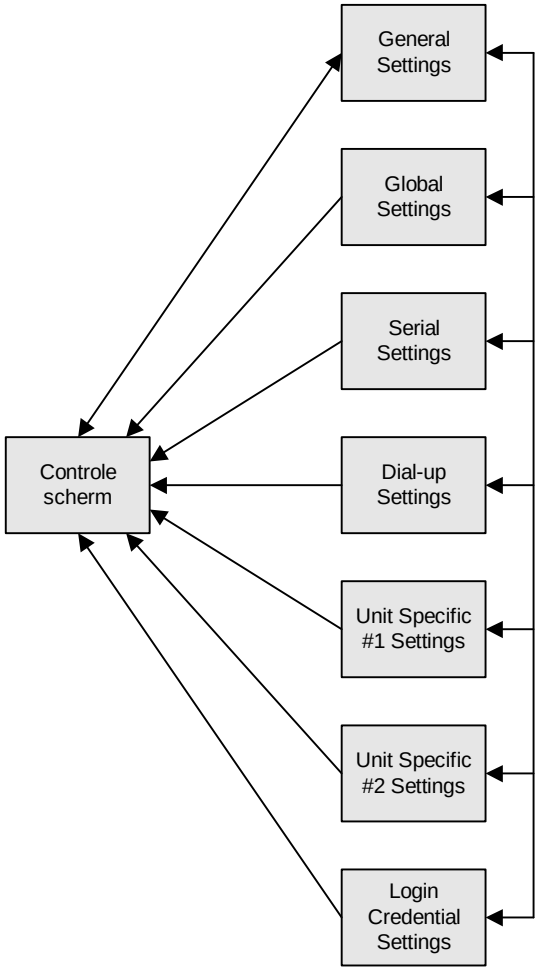
4.1.1 Userinterface Concept Basisontwerp

Afhankelijk van de functie van de gebruiker wordt een keuze gemaakt wat er gedaan moet worden. Standaard wordt de controle module voor de controleur gestart. Een beheerder kan er voor kiezen hiervandaan de beheer module uit te voeren.

De beheer module toont de gebruiker, in een boomstructuur, alle units en unit-groepen. De beheerder kan er dan voor kiezen om de connectie gegevens van een unit(groep) te bekijken door er op te klikken. Ook kan hij ervoor kiezen een unit(groep) te verwijderen of te wijzigen.

De controle module toont de gebruiker, in een boomstructuur, alle units en unit-groepen met daarachter de status van de unit(groep). Daarnaast is er een overzicht te zien met alle verwachte acties (events). Hierin kan de controleur bijvoorbeeld zien dat er een ad-insert verwacht wordt op een bepaald tijdstip. Daarnaast is er een lijst te zien met alle laatste geconstateerde fouten en een kaart van Nederland met daarop de locaties waar een unit(groep) met problemen zich bevind.

4.1.2 Pagina Navigatie Eisen



Figuur 4-1 Pagina navigatie verloop

5 GEDETAILEERDE EISEN

Uitgaande van de functionaliteiten die op hoger niveau zijn beschreven in de UI specificatie en het use case model, worden in dit hoofdstuk de gedetailleerde eisen op een laag niveau beschreven. Dit hoofdstuk is niet bedoeld als een stuk dat eenvoudigweg van voor tot achter wordt gelezen, maar eerder als een referentie om gedetailleerde eisen te specificeren.

De gedetailleerde eisen specificatie bevat alle eisen die binnen het systeem van belang zijn en wordt gebruikt en gestructureerd als een referentiehandleiding. De eisen worden gegroepeerd in twee categorieën; functionele eisen en kwaliteitseisen. De functionele eisen worden beschreven in termen als ‘het systeem zal X gaan doen’. Binnen de kwaliteitseisen wordt beschreven ‘hoe goed doet het systeem functie X’. Binnen bovengenoemde categorieën worden de eisen verder onderverdeeld op de volgende manier.

Functionele eisen	Kwaliteitseisen
Datatype specificaties	Beveiliging
Business regels	Performance
Userinterface eisen	Robuustheid
Business automatisering	Concurrency
Rapportages	Internationale mogelijkheden
Systeem interfaces	Configuratie
Systeem administratie	
Installatie	
Migratie	
Operationeel	

5.1 Eisen Identificatienummers

Aan iedere eis wordt een unieke identifier toegewezen, op basis van de volgende regels:

Nr.	Identifier	Beschrijving
1.	DR-ss-nn-mmm	Data formatting requirement
2.	BR-ss-nn-mmm	Business rule requirement
3.	FR-ss-nn-mmm	Functional requirement
4.	IR-ss-nn-mmm	Interface requirement
5.	PR-ss-nn-mmm	Proficiency requirement (beveiling, performance enz.)

Hierbij is ss het sectienummer waarin de eis verschijnt, nn is de sequentiële nummering van de eisen binnen de sectie en mmm wordt gebruikt om eisen in te voegen zonder dat andere eisen moeten worden hernummerd.

Formulieren en rapporten worden niet aangemerkt als zijnde eisen, maar in plaats daarvan worden er unieke identifiers aan toegewezen.

Nr.	Identifier	Beschrijving
1.	Form-nnnn-mmm	Formulier nummer en identifier
2.	Report-nnnn-mmm	Rapportage nummer en identifier

5.2 Systeem Functionaliteiten Ondersteund in APM

De gebruiker kan in het systeem units en unit-groepen definiëren, uitlezen wat de units geprogrammeerd hebben en staan en controleren of de units de geprogrammeerde schema's ook correct hebben uitgespeeld.

5.3 Systeem Functionaliteiten Niet Ondersteund in APM

Op dit moment zal het systeem geen ondersteuning bieden voor het real-time ingrijpen in het uitspeel proces. Daarnaast zal het systeem nog geen ondersteuning bieden voor meerdere gebruikers en gescheiden rechten.

5.4 Functionele Eisen

Het doel van de gedetailleerde eisen beschrijving is het specificeren van de kern functies van het systeem; 'wat gaat het systeem precies doen' . De volgende sectie specificeert vervolgens hoe goed het deze functies uit zal moeten voeren. De functionele eisen in deze sectie worden gegroepeerd in de volgende subsecties.

- Architectuur en ontwerp begrenzingen;
- System Level Business rules eisen;
- Procesautomatiseringseisen;
- User interface eisen;
- Rapporten;
- Systeem Interface eisen;
- Systeem administratie eisen;
- Installatie eisen;
- Migratie eisen;
- Operationele eisen;
- Datatype specificaties.

5.4.1 Architectuur en Ontwerp Begrenzingen

Het systeem moet flexibel zijn. Modules moeten onafhankelijk van elkaar kunnen draaien op aparte werkstations. Er zal één computer ingericht worden als server, deze onderhoud het contact met de units. Vanaf elke willekeurige computer met een verbinding met de server moeten daarna de beheer en controle modules gedraaid kunnen worden.

5.4.2 System Level Business Rules Eisen

In deze paragraaf worden de business rules (regels betreffende procedures die gebruikt worden binnen het systeem) op het hoogste niveau beschreven die automatisch door het systeem worden gecontroleerd en die ook vereist zijn bij bepaalde acties binnen het systeem. Andere eisen in dit document refereren aan

deze business rules, wat aangeeft wanneer deze regels worden vereist binnen het systeem en hoe de externe actoren worden ingelicht wanneer niet aan de business rule wordt voldaan.

RequirementID	Rule
BR-32-01-0001	NetworkSettings en SerialSettings kunnen alleen gedefinieerd worden als er een Entity gecreëerd is.
BR-32-01-0002	DirectSettings en DialUpSettings kunnen alleen gedefinieerd worden als er een SerialSetting voor gecreëerd is.
BR-32-01-0003	LoginCredentials en XcpSettings kunnen alleen gedefinieerd worden als er een NetworkSetting gecreëerd is.
BR-32-01-0004	TelnetSettings en FtpSettings kunnen allen gedefinieerd worden als er een LoginCredential gedefinieerd is.
BR-32-02-0000	Name mag in iedere groep of subgroep slechts één keer voorkomen.

5.4.3 Procesautomatiserings Eisen

Geen.

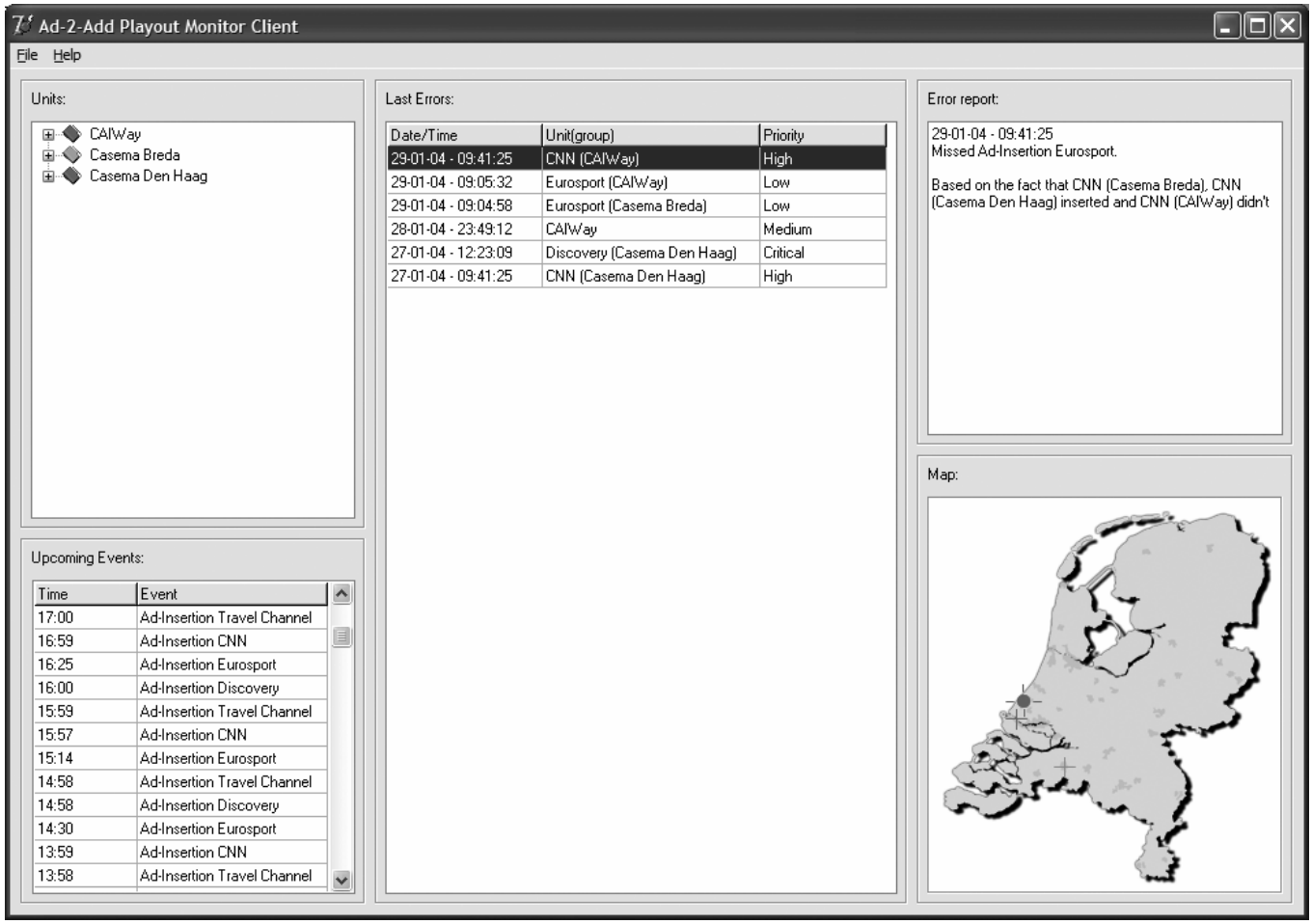
5.5 Userinterface Eisen

In deze paragraaf wordt de UI Specificatie uitgelegd en uitgebreid en wordt zodoende gespecificeerd wat de gedetailleerde eisen zijn van de UI van dit systeem. Waar in het UI ontwerp en UI Prototype het algemene uiterlijk, de structuur en de schermvolgorde van de userinterface wordt beschreven, worden in deze paragraaf de gedetailleerde eisen van ieder frame van de interface beschreven.

5.5.1 Algemene eisen

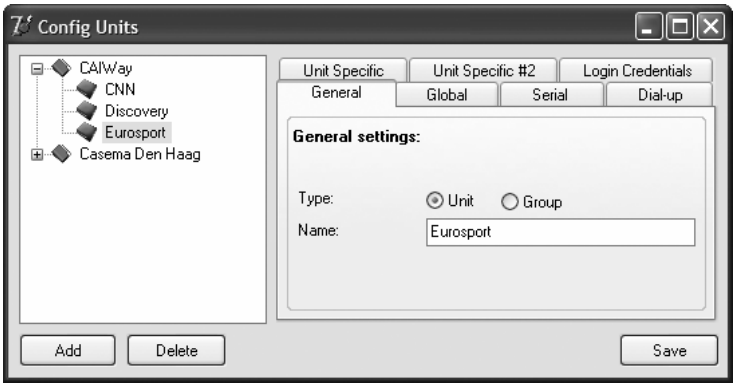
RequirementID	Rule
FR-341-0001-000	In de configuratie schermen kunnen alleen connectie gegevens ingevoerd worden als het verloop conform BR-32-01-0001 t/m BR-32-01-0004 is.

5.5.2 Controle formulier (Form-0001-000)



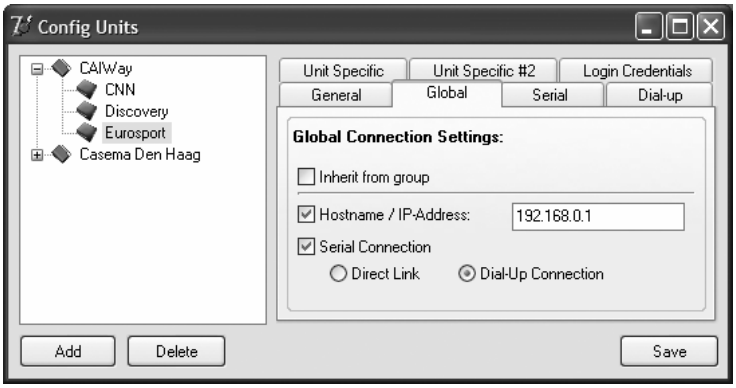
Figuur 5-1 Play-out monitor client UI prototype

5.5.3 Unit beheer formulier - General (Form-0002-0001)



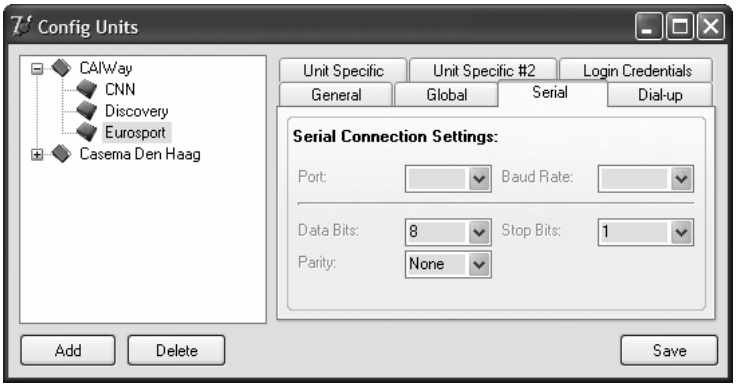
Figuur 5-2 Unit manager general settings UI prototype

5.5.4 Unit beheer formulier – Global (Form-0002-0002)



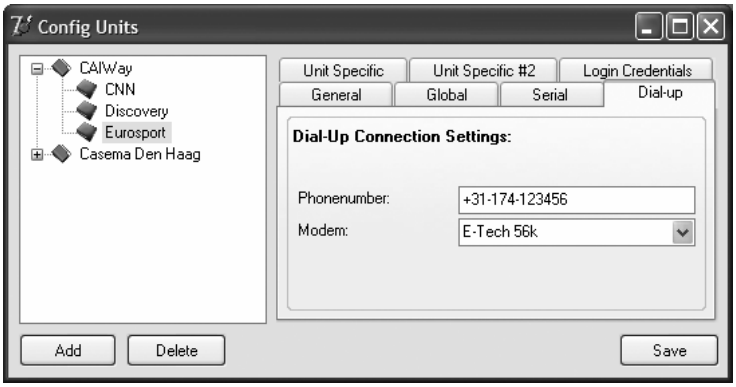
Figuur 5-3 Unit manager global settings UI prototype

5.5.5 Unit beheer formulier – Serial (Form-0002-0003)



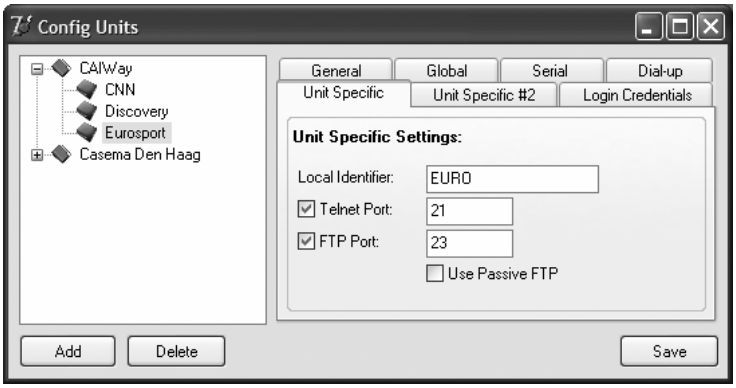
Figuur 5-4 Unit manager serial settings UI prototype

5.5.6 Unit beheer formulier – Dial-up (Form-0002-0004)



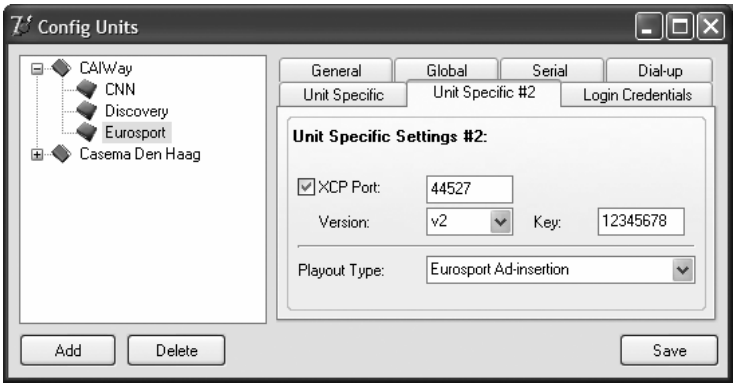
Figuur 5-5 Unit manager dial-up settings UI prototype

5.5.7 Unit beheer formulier – Unit Specific #1 (Form-0002-0005)



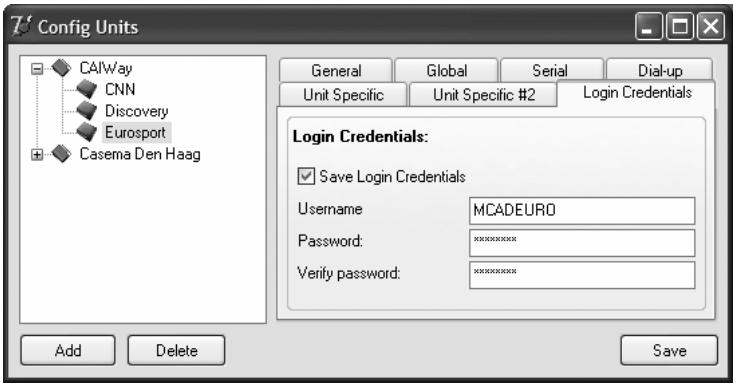
Figuur 5-6 Unit manager unit specific #1 settings UI prototype

5.5.8 Unit beheer formulier – Unit Specific #2 (Form-0002-0006)



Figuur 5-7 Unit manager unit specific #2 settings UI prototype

5.5.9 Unit beheer formulier – Login Credentials (Form-0002-0007)



Figuur 5-8 Unit manager login credentials UI prototype

5.6 Kwaliteitseisen

In deze paragraaf worden de kwaliteitseisen die gesteld worden aan het systeem beschreven.

5.6.1 Beveiliging Eisen

Omdat APM in eerst instantie in een vertrouwde omgeving ingezet zal worden en met het systeem geen invloed uitgeoefend kan gaan worden op het uitspelen worden er geen eisen gesteld aan authenticatie. Om te voorkomen dat kwaadwillende gebruikers die toegang hebben verkregen tot het systeem unit login gegevens kunnen raadplegen zullen deze wel beschermd moeten worden. Er zullen ook eisen gesteld worden aan data integriteit. Alle data moet te allen tijde correcte zijn.

RequirementID	Rule
PR-413-0001-000	Login gegevens om de units te benaderen moeten gecodeerd opgeslagen worden.

5.6.2 Performance Eisen

De performance moet van dien aard zijn dat de gebruiker geen lange wachttijden (langer dan ± 1 seconde) mag ervaren bij het ophalen van uitspeel gegevens. Tevens mag het aantal unit-polls niet beïnvloed worden door de grote van de database.

5.6.3 Robuustheid Eisen

In onderstaande tabel zijn een aantal eisen vastgelegd betreffende beschikbaarheid van APM.

RequirementID	Rule
PR-43-0001-001	De APM server module moet 24/7 beschikbaar zijn.
PR-43-0001-002	De downtime van de server module mag maximaal 5 uur per maand zijn.

5.6.4 Concurrency Eisen

In onderstaande tabel zijn de eisen beschreven die betrekking hebben op concurrency binnen het systeem.

RequirementID	Rule
PR-44-0001-000	Optimistic concurrency wanneer 2 records gelijktijdig worden geüpdate.

5.6.5 Internationalisatie Eisen

APM wordt met het Engels als voertaal ontwikkeld. Er zijn geen plannen voor verdere internationalisatie.

5.6.6 Configuratie Eisen

In onderstaande tabel zijn de eisen beschreven die betrekking hebben op de configuratie het systeem.

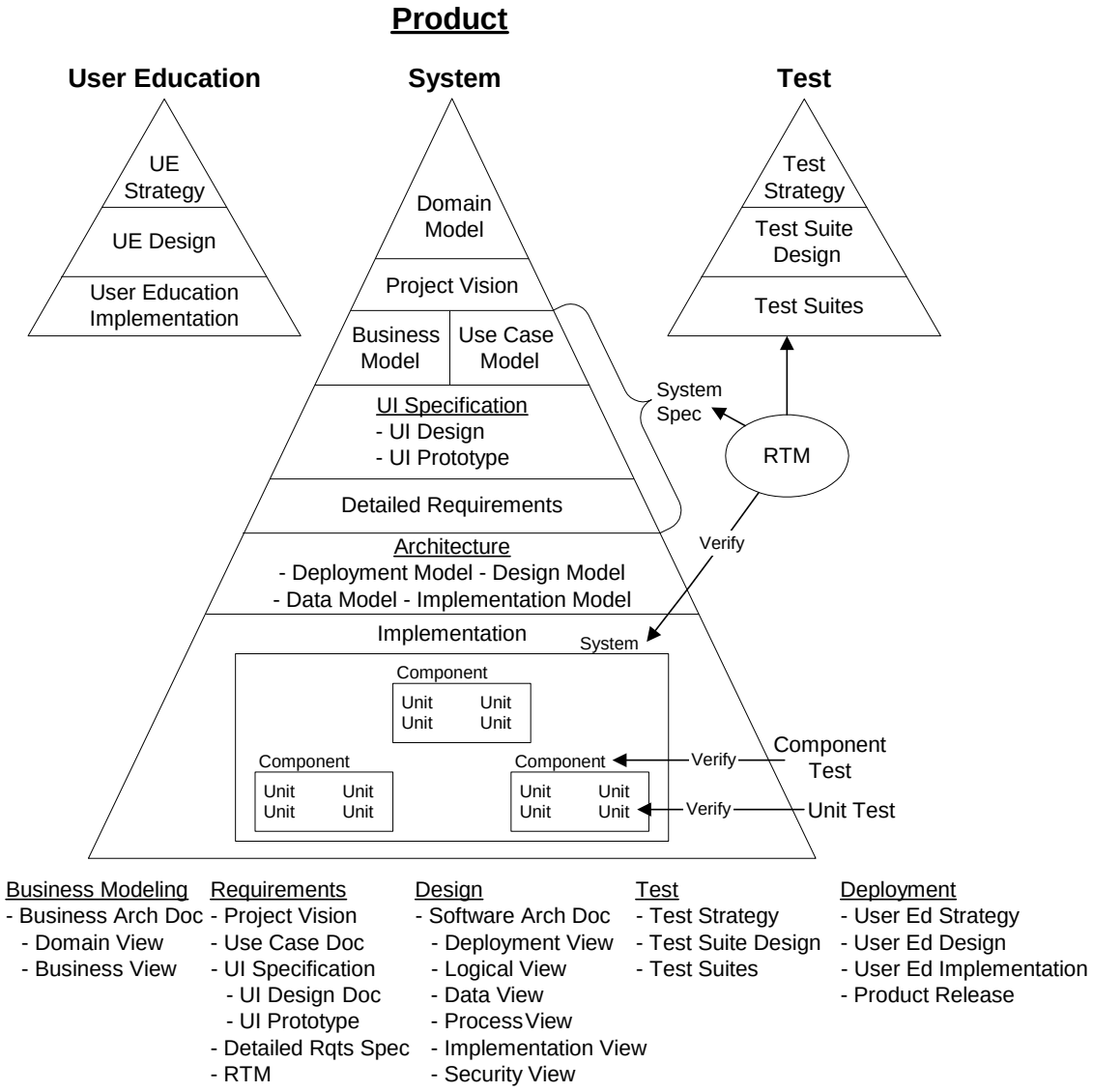
RequirementID	Rule
PR-46-0001-000	De poll interval moet instelbaar zijn. Zowel de standaard interval als de ‘hot-interval’ (moment dat een event verwacht wordt op een bepaald kanaal).

BIJLAGE D.
SOFTWARE ARCHITECTURE

INHOUDSOPGAVE

1 INLEIDING

In het Software Architecture Document worden alle interne beslissingen beschreven die nodig zijn om het systeem te implementeren.



Figuur 1-1 RUP producten en hun relaties

De architectuur van een software systeem vereist zes verschillende views, waarbij iedere view zich richt op een ander aspect van het systeem. Het doel is te beschrijven wat de belangrijkste onderdelen van het systeem zijn, hoe het is gestructureerd, wat de systeem proces stromen zij en wat de belangrijkste interfaces zijn. Vanuit een hoger niveau is het doel om inzicht te krijgen in het systeem vanuit verschillende perspectieven en daardoor alle kritieke systeem functionaliteiten in kaar te brengen.

Deployment View: Deze view documenteert de fysieke topologie van het systeem zoals gemodelleerd in het Deployment Model. Er wordt in beschreven hoe computers worden geïmplementeerd en hoe de connectie tussen deze computers is. De configuratie van ieder onderdeel wordt ook beschreven; operations system, database, COTS (commercial off the shell = materiaal beschikbaar op de markt) en programmatuur.

Logical View: De logical view documenteert het Design Model, waarin de layers binnen de applicatie worden gedefinieerd inclusief de primary classes behorende bij iedere layer. De systeem architect identificeert patronen in de functionaliteit en creeërt mechanismen die in deze functionaliteit voorzien in verschillende onderdelen van de applicatie.

Data View: Klassen in de logical view worden geclassificeerd als zijnde transient of persistent. De persistente klassen worden in kaart gebracht in structuren op disk, meestal in een combinatie van rijen in een relationele database. Een entity-relationship model beschrijft het database schema. Deze view brengt ook in kaart hoe de objectgeoriënteerde klassen in de relationele tabellen staan.

Process (Concurrency) View: Deze view richt zich het concurrency aspect van het systeem en hoe er wordt omgegaan met gedeelde bronnen/ hulpmiddelen. De process view documenteert de onafhankelijke communicatiesystemen binnen het systeem en beschrijft ook hoe deze communiceren. Daarnaast geeft het een overzicht van de bronnen die door deze lijnen worden gebruikt en het transactiemodel dat wordt gebruikt voor het behouden van de integriteit van deze bronnen.

Implementation View: Binnen deze view wordt aangegeven welke klassen uit de Logical View horen bij welke fysieke bronbestanden en combineert de bestanden in werkbare componenten. De Implementation view houdt ook bij welke afhankelijkheden er bestaan tussen componenten.

Security View: Deze view richt zich op hoe gebruikers zich identificeren, hoe rechten worden toegekend op basis van identiteit, hoe zorg wordt gedragen voor de integriteit van het systeem en de data en voor het monitoren en loggen van activiteiten.

2 ARCHITECTUUR RICHTLIJNEN

Het systeem zal worden gebouwd volgens de volgende richtlijnen:

- Objectgeoriënteerde ontwerp principes. Business objecten die ontdekt zijn tijdens de analyse vormen de basis van de business services API. Attributen en gedrag samengevat binnen business klassen.
- De component view is gebaseerd op drie lagen; UI (userinterface) Laag, de Business laag en de Data laag. De UI laag bindt attributen van business objecten tot UI objecten. Vervolgens worden deze gepresenteerd aan de gebruiker, wordt input geaccepteerd en worden opdrachten verwerkt door het ontbinden van UI objecten tot attributen van business objecten. De business laag checkt business regels en geeft business objecten door aan de data laag. De datalaag zet business attributen om naar relationele tabellen en omvat dit proces in één transactie.
- Minimaliseer de afhankelijkheid van een bepaalde tool of een bepaalde leverancier.
- Maximaliseer de invoering van beproefde standaarden en technologieën.
- Ontwerp voor het op langere termijn toepassen van principes als:
    - Onderhoudbaar vanaf één plaats
    - Configureerbaar, uitbreidbaar
    - Schaalbaar
- Meerdere ontwerp patronen zijn samengevat in herbruikbare componenten. Deze patronen zijn:
    - Create/Read/Update/Delete onderdelen van business objecten.
    - UI sessie stapel modellen (zoals beschreven in het GUI ontwerp document)
    - Verbinden en ontbinden van attributen van business objecten aan UI objecten.
    - Vereiste, maximale lengte, string validatie voor UI input velden.
    - Optimistic concurrency

2.1 User Interface laag richtlijnen

In deze paragraaf worden de richtlijnen beschreven waarop de userinterface laag wordt gebaseerd.

- De UI zal verborgen en onverborgen edit checks uitvoeren (bijvoorbeeld controle op correcte invoer), controle op maximum lengte en validatie van vereiste attributen. Dit alles wordt gebaseerd op een meta data interface die geëxporteerd wordt vanuit de business laag.
- Pagina's worden volgens een modulaire opbouw ontworpen, waarbij dezelfde principes worden gevolgd als wanneer een klasse wordt ontworpen.

2.2 Business laag richtlijnen

In deze paragraaf worden de richtlijnen bij het ontwerp van de business laag beschreven.

- De business laag geeft waarden van business objecten terug. De grootste verantwoordelijkheid van de business laag is om de illusie te geven aan de UI laag dat alle business objecten in het fysieke geheugen staan.
- Dit houdt tevens in dat de business laag interface geen informatie over het DB schema prijsgeeft.
- De business laag voorziet in een basis klasse voor een geverifieerde gebruiker. Dit verifiëren is de verantwoordelijkheid van de UI laag. Aan deze gebruiker worden alle aanroepen aan data services doorgegeven.
- De business laag ondersteunt het “dirty bit” ontwerp patroon. Objecten worden alleen gewijzigd indien er ook daadwerkelijk iets is veranderd.
- Er zijn twee typen business objecten:
    - Reference objecten (stabiele, read-only objecten die worden gecached. Alle sessies refereren aan een enkele instantie van deze klassen.
    - Business objecten (instanties worden opgenomen binnen de strekking van een sessie.)
- De business laag voert business regel checks uit voordat data wordt opslagen in de database.
- De business laag voorziet ondersteuning voor optimistic concurrency doormiddel van een versie timestamp in het hoofd business object.

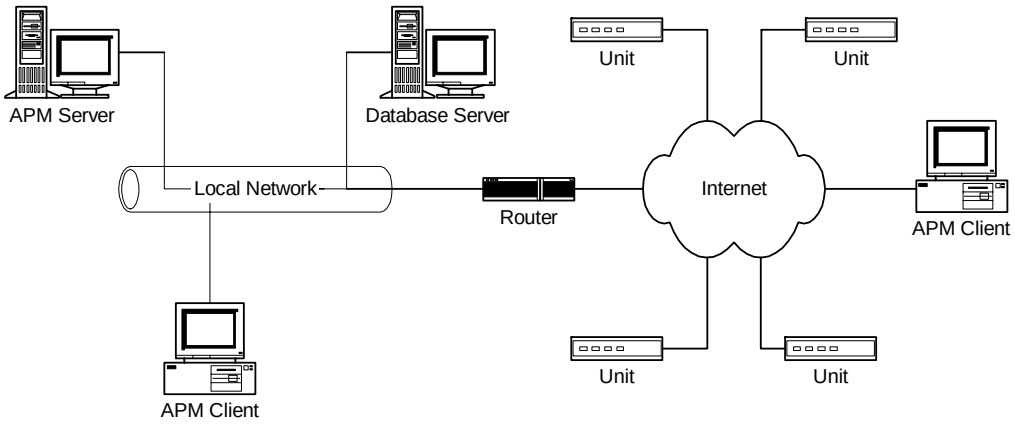
2.3 Data laag richtlijnen

In deze paragraaf worden de richtlijnen voor de datalaag beschreven.

- De datalaag omvat volledig het indelen van business attributen naar relationele tabellen. Dit maakt het mogelijk dat de database wordt herontworpen zonder dat business of gebruiker services worden aangepast.
- Invloed op referentiële integriteit, uniek, null/not null eisen.
- Het creëren van een interne unieke primary key voor iedere tabel. De belangrijkste reden hiervoor is dat, als de primary key wordt gebruikt als publiekelijk zichtbare identifier, het niet mogelijk is om het hernoemen van de unieke ID te ondersteunen.

3 DEPLOYMENT VIEW

In de deployment view wordt de topologie en de fysieke en logische connecties van het netwerk weergegeven. Daarnaast wordt gedetailleerd aangegeven wat de configuratie is van ieder onderdeel van het netwerk. De afkortingen zijn uitgelegd in de tabel op de volgende pagina.



Figuur 3-1 Fysieke en logische verbindingen die samen het APM systeem vormen

3.1 APM server

In onderstaande tabel staan de specificaties die benodigd zijn voor de APM server gedefinieerd.

Software	Versie	Leverancier	Opmerkingen
OS	Windows XP Server		

In de volgende tabel zijn de eisen gespecificeerd die worden gesteld aan de hardware die deel uitmaakt van de APM server.

Component	Specificaties	Leverancier	Opmerkingen
Processor	Pentium 4 2.80 Ghz		
Geheugen	512 Mb		
Harddisk	20 Gb		
DVD-Rom	16x 48x		
NIC	100 Mbit		

3.2 Database server

In onderstaande tabel staan de specificaties die benodigd zijn voor de database server gedefinieerd.

Software	Versie	Leverancier	Opmerkingen
OS	Debian Linux 3.0		
Database	MySQL 4.1		

In de volgende tabel zijn de eisen gespecificeerd die worden gesteld aan de hardware die deel uitmaakt van de APM server.

Component	Specificaties	Leverancier	Opmerkingen
Processor	Pentium 4 2.80 Ghz		
Geheugen	512 Mb		
Harddisk	20 Gb		
DVD-Rom	16x 48x		
NIC	100 Mbit		

3.3 Client workstation

In onderstaande tabel staan de specificaties die benodigd zijn voor de APM server gedefinieerd.

Software	Versie	Leverancier	Opmerkingen
OS	Windows 2000 of hoger		

In de volgende tabel zijn de eisen gespecificeerd die worden gesteld aan de hardware die deel uitmaakt van de APM server.

Component	Specificaties	Leverancier	Opmerkingen
Processor	Pentium 3 of hoger		
Geheugen	128 Mb of hoger		
Harddisk	15 Mb of meer		
NIC	10 Mbit of sneller		

3.4 Project software

Computer	Component	Installatie parameters
APM server	APM Server	Verwijzend naar de database op de database server
Database server		Open voor queries vanaf de APM server.
Client workstation	APM Client	Verwijzend naar de APM server.

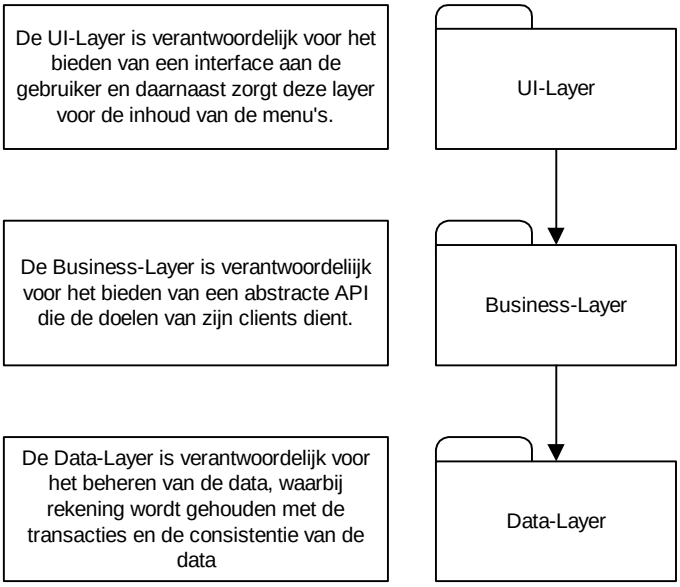
4 LOGICAL VIEW

De logical view representeert de kern van het ontwerp van het systeem. Het geeft de primaire klassen weer die samenwerken om de systeem functionaliteiten te implementeren. De logical view beval de volgende onderdelen:

- Drie software layers
- Userinterface session stack
- Cached application reference objects

4.1 Software layers

De applicatie wordt gestructureerd doormiddel van drie aparte layers: de UI layer, de Business Layer en de Data Layer. In onderstaande figuur zijn de drie verschillende lagen weergegeven inclusief een beschrijving van de inhoud en betekenis per laag.



Figuur 4-1 Drie software lagen

UI Layer: Verantwoordelijk voor de authenticatie, autorisatie en de presentatie. De schermlay-out is gedefinieerd in de applicatie. Data die wordt ingevoerd (bijvoorbeeld administratieve gegevens) worden zowel lokaal (tijdelijk, totdat alles wordt weggeschreven naar de database) als in de database opgeslagen.

Business Layer: In de business layer (ook wel middleware layer genoemd) wordt de logica achter de systeemprocessen beschreven. Deze laag zorgt voor de interactie met andere (gedistribueerde) middleware technologieën. Binnen APM zal gebruik worden gemaakt van MySQL.

Data Layer: Verantwoordelijk voor het managen van data en transacties. Brengt in kaart waar objecten uit het systeem fysiek zijn opgeslagen (in welke tabellen).

Daarnaast is deze layer verantwoordelijk voor de integriteit van de data en data intensieve systeem regels zoals unieke namen bij kolommen.

4.2 Userinterface Session Stack

Binnen APM is er geen moment waarop een session stack bijgehouden zal worden. Door het feit dat er in ieder geval geen sprake is van inlog procedures vervalt de stack welke account gegevens e.d. bijhoudt. Daarnaast wordt het gebruik van APM ook niet afgehandeld in de vorm van sessies. Er kan altijd ‘ingeprikt’ worden om de huidige unit statussen uit te lezen.

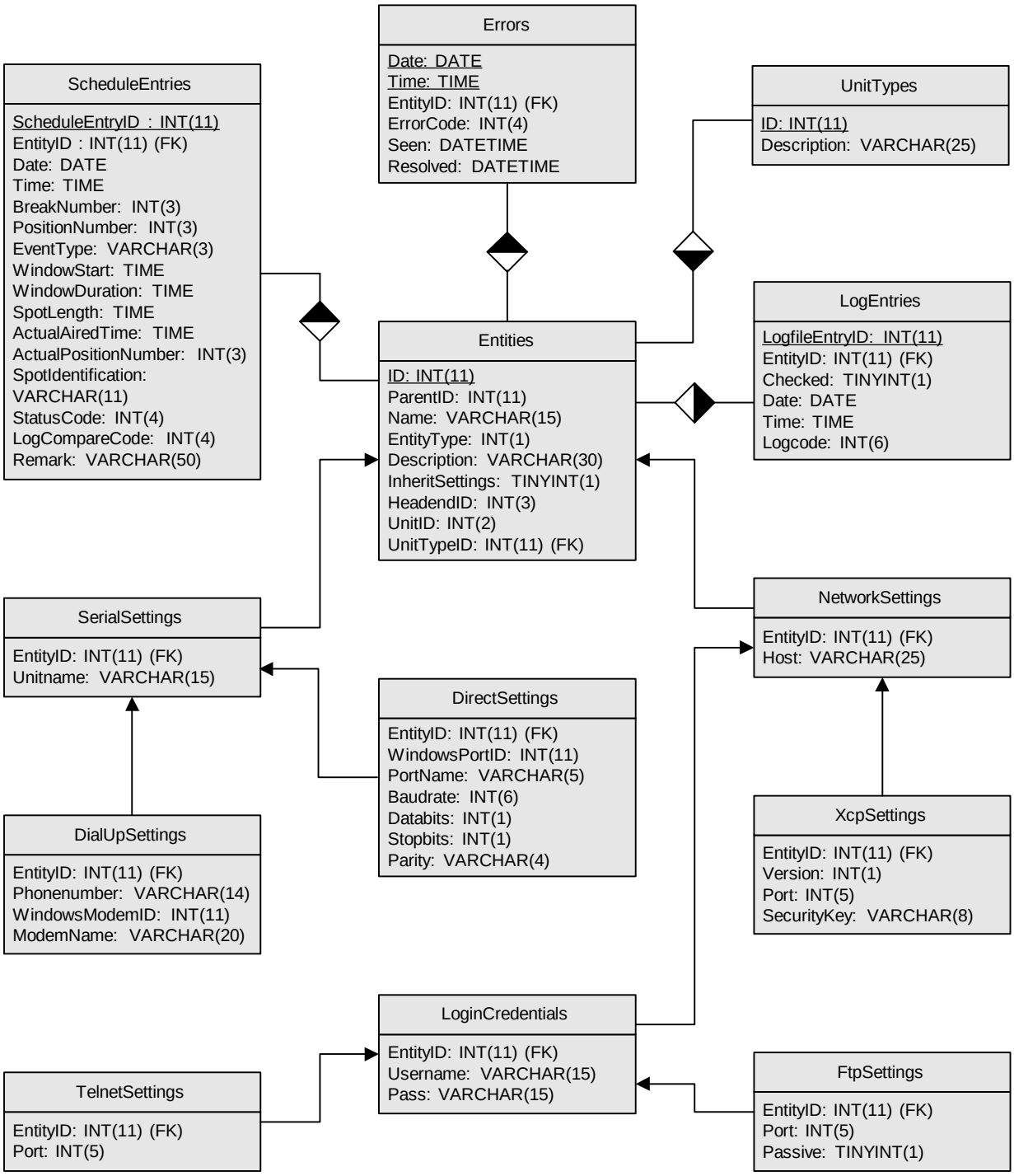
4.3 Cached Application Reference Objects

Uitsluitend de data die de verschillende uittypen beschrijft wordt eenmalig ingelezen en opgeslagen in het geheugen wanneer de gebruiker er voor kiest om nieuwe units of headends in te voeren in het systeem.

5 DATA VIEW

Het database model wordt gepresenteerd als een logical view, een fysieke view en een datadictionary. Binnen APM wordt gebruik gemaakt van twee verschillende databases; de units database en de playout database. Voor allebei deze databases worden een ERD-model gemaakt.

5.1 Logical View



Figuur 5-1 Logical View op de database

5.2 Physical View

```
CREATE TABLE unittypes (
      ID INT(11) NOT NULL AUTO_INCREMENT,
      Description VARCHAR(25) NOT NULL,
      PRIMARY KEY(ID)
)

CREATE TABLE entities (
      ID INT(11) NOT NULL AUTO_INCREMENT,
      ParentID INT(11) NOT NULL,
      UnitTypeID INT(11) NULL,
      Name VARCHAR(15) NOT NULL,
      EntityType INT(1) NOT NULL,
      Description VARCHAR(30) NULL,
      InheritSettings TINYINT(1) NOT NULL,
      HeadendID INT(3) NULL,
      UnitID INT(3) NULL,
      PRIMARY KEY(ID)
      FOREIGN KEY(ParentID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
      FOREIGN KEY(UnitTypeID)
            REFERENCES unittypes(ID)
                  ON DELETE RESTRICT
                  ON UPDATE NO ACTION
)

CREATE TABLE scheduleentries (
      ScheduleEntryID INT(11) NOT NULL AUTO_INCREMENT,
      EntityID INT(11) NOT NULL,
      Date DATE NOT NULL DEFAULT 0000-00-00,
      Time TIME NOT NULL DEFAULT 00:00:00,
      BreakNumber INT(3) NOT NULL,
      PositionNumber INT(3) NOT NULL,
      EventType VARCHAR(3) NOT NULL,
      WindowStart TIME NOT NULL DEFAULT 00:00:00,
      WidowDuration TIME NOT NULL DEFAULT 00:00:00,
      SpotLength TIME NOT NULL DEFAULT 00:00:00,
      ActualAiredTime TIME NULL,
      ActualPositionNumber INT(3) NULL,
      SpotIdentification VARCHAR(11) NOT NULL,
```

```
      StatusCode INT(4) NULL,
      LogCompareCode INT(4) NULL,
      Remark VARCHAR(50) NULL,
      PRIMARY KEY(ScheduleEntryID, EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)

CREATE TABLE networksettings (
      EntityID INT(11) NOT NULL,
      Host VARCHAR(25) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)

CREATE TABLE serialsettings (
      EntityID INT(11) NOT NULL,
      Unitname VARCHAR(15) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)

CREATE TABLE xcpsettings (
      EntityID INT(11) NOT NULL,
      Version INT(11) NOT NULL,
      Port INT(5) NOT NULL,
      SecurityKey VARCHAR(8) NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES networksettings(EntityID)
                  ON DELETE CASCADE
```

```
                ON UPDATE CASCADE
)

CREATE TABLE logincredentials (
      EntityID INT(11) NOT NULL,
      Username VARCHAR(15) NOT NULL,
      Pass VARCHAR(15) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES networksettings(EntityID)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
)

CREATE TABLE directsettings (
      EntityID INT(11) NOT NULL,
      WindowsPortID INT(11) NOT NULL,
      PortName VARCHAR(5) NOT NULL,
      Baudrate INT(6) NOT NULL,
      Databits INT(1) NOT NULL,
      Stopbits INT(1) NOT NULL,
      Parity VARCHAR(4) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES serialsettings(EntityID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)

CREATE TABLE dialupsettings (
      EntityID INT(11) NOT NULL,
      Phonenumber VARCHAR(14) NOT NULL,
      WindowsModemID INT(11) NOT NULL,
      ModemName VARCHAR(20) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES serialsettings(EntityID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
)
```

```
CREATE TABLE errors (
      Date DATE NOT NULL DEFAULT 0000-00-00,
      Time TIME NOT NULL DEFAULT 00:00:00,
      EntityID INT(11) NOT NULL,
      Errorcode INT(4) NOT NULL,
      Seen DATETIME NOT NULL DEFAULT 0000-00-00 00:00:00,
      Resolved DATETIME NOT NULL DEFAULT 0000-00-00 00:00:00,
      PRIMARY KEY(Date, Time, EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)

CREATE TABLE logentries (
      LogfileEntryID INT(11) NOT NULL AUTO_INCREMENT,
      EntityID INT(11) NOT NULL,
      Checked TINYINT(1) NOT NULL DEFAULT 0,
      Date DATE NOT NULL DEFAULT 0000-00-00,
      Time TIME NULL,
      Logcode INT(6) NULL,
      PRIMARY KEY(LogfileEntryID, EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES entities(ID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)

CREATE TABLE ftpsettings (
      EntityID INT(11) NOT NULL,
      Port INT(5) NOT NULL,
      Passive TINYINT(1) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES logincredentials(EntityID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)
```

```
CREATE TABLE telnetsettings (
      EntityID INT(11) NOT NULL,
      Port INT(5) NOT NULL,
      PRIMARY KEY(EntityID),
      INDEX EntityID(EntityID),
      FOREIGN KEY(EntityID)
            REFERENCES logincredentials(EntityID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
)
```

6 PROCESS VIEW

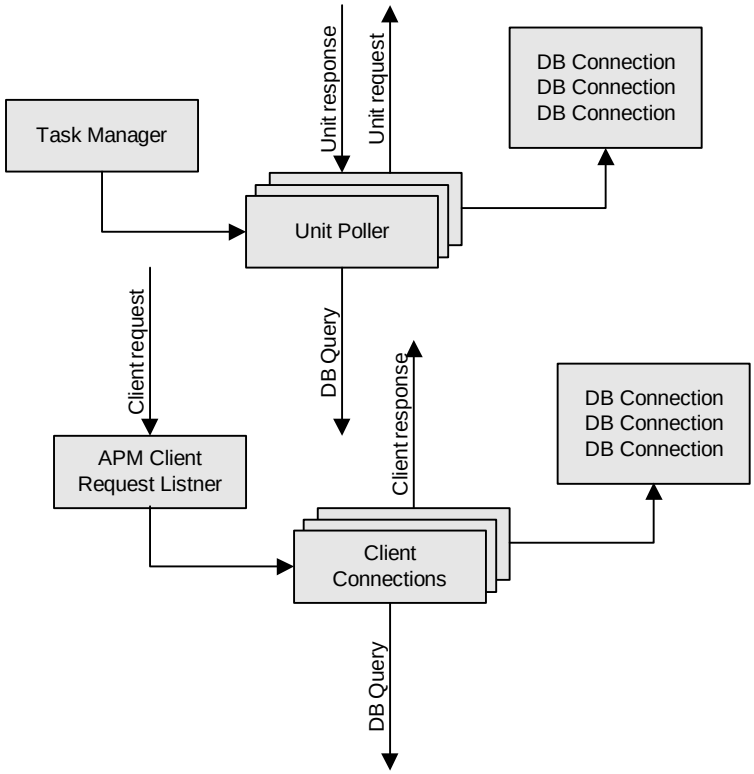
Binnen deze view wordt vooral gekeken naar de parallelle processen die plaatsvinden binnen het systeem.

Binnen APM onderscheiden we twee vormen van concurrency:

- CPU/process/threads ontwerp
- Database transacties

6.1 CPU/process/threads ontwerp

APM Server zal zo opgebouwd worden dat er meerdere threads tegelijkertijd gecreëerd worden om data van de units op te halen en dat er meerdere threads tegelijkertijd gecreëerd worden om de data verzoeken van de APM clients af te handelen.



Figuur 6-1 APM Server concurrency

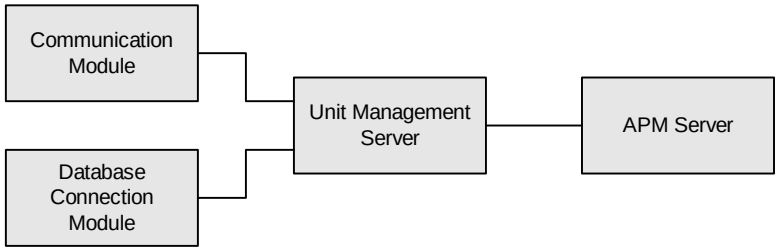
6.2 Database transacties

Alle transacties worden uitgevoerd door een aanroep vanuit de Business laag van een stored procedure. Eventuele SQL fouten zullen dan een rollback tot gevolg hebben en een foutmelding retourneren aan de Business laag.

In principe zal de task manager in APM Server er voor zorgen dat alle threads unieke en relevante data opvragen bij de units en invoeren in de database. Voor de zekerheid zal er ook gebruik gemaakt worden van optimistic concurrency om de data-integriteit te waarborgen. Bij iedere update wordt allereerst gecontroleerd of de in te voeren data daadwerkelijk nieuwe data betreft. Dit gebeurd door eerst te controleren wat de datum en tijd van de laatst ingevoerde logentries is en deze te vergelijken met de datum en tijd van de nieuw in te voeren logentries.

7 IMPLEMENTATION VIEW

APM zal bestaan uit twee gedeeltes, een server en een client gedeelte. De client zal bestaan uit één geheel, de server zal bestaan uit meerdere gedeeltes van herbruikbare componenten.



Figuur 7-1 Server Implementatie model

Er is gekozen voor meerdere componenten, omdat de verwachting bestaat dat in de toekomst het systeem uitgebreid zal worden met nieuwe diensten. De Unit Management Server fungeert in dit geheel dan als component wat alle diensten (zoals APM Server) in één systeem integreert. De diensten kunnen gebruik maken van de modules van de unit management server om verbinding te maken met units en databases. Deze modules zijn opgesplitst, omdat in de toekomst deze mogelijk vervangen of uitgebreid kunnen worden met nieuwe of andere functionaliteiten. Mogelijke veranderingen in de modules kunnen zijn: updates in het XCP protocol en/of de keuze voor een ander DBMS.

Uitgaande van de eerder gedefinieerde software lagen zal APM als volgt opgedeeld worden:

UI Layer:

- APM Client (Errors, Entities)

Business Layer:

- APM Server (Schedules, ScheduleEntries, LogFiles, LogEntries, Errors)
- Unit Management Server (Entities)
- Communication Module (SerialSettings, NetworkSettings, DialUpSettings, DirectSettings, LoginCredentials, XcpSettings, TelnetSettings, FtpSettings)
- Database Communication Module (Schedules, ScheduleEntries, LogFiles, LogEntries, Erros, Entities, SerialSettings, NetworkSettings, DialUpSettings, DirectSettings, LoginCredentials, XcpSettings, TelnetSettings, FtpSettings)

Data Layer:

- Database tabellen
- Database Connection Module

8 SECURITY VIEW

Binnen de security is beschreven hoe binnen het systeem de beveiligingseisen zijn geïmplementeerd. Het ontwerp van de beveiliging wordt gepresenteerd doomiddel van de volgende subonderwerpen.

- Gebruiker identificatie en authenticatie – Hoe identificeert het systeem gebruikers en controleert of het de betreffende gebruiker is.
- Autorisatie – Zodra de authenticatie heeft plaatsgevonden, wie mag wat doen binnen het systeem.
- Data integriteit en privacy – Verzekeren dat de integriteit van de data niet wordt aangetast.
- Non-repudiation en auditing – Ervoor zorgen dat de eindgebruikers hun acties binnen het systeem niet kunnen verwijderen. Bewaren van gegevens over het gebruik van de bestanden zodat bekend is wie wat heeft gedaan.

8.1 Gebruiker identificatie en authenticatie

Binnen APM zal in eerste instantie geen gebruik worden gemaakt van identificatie en authenticatie.

8.2 Autorisatie

Binnen APM zal in eerste instantie geen gebruik worden gemaakt van autorisatie.

8.3 Data integriteit en privacy

Paswoorden die gebruikt worden om contact te maken met de units worden encrypted voordat ze worden opgeslagen in de database. Tevens wordt het paswoord wat wordt gebruikt om verbinding te maken met de database encrypted opgeslagen in het registery.

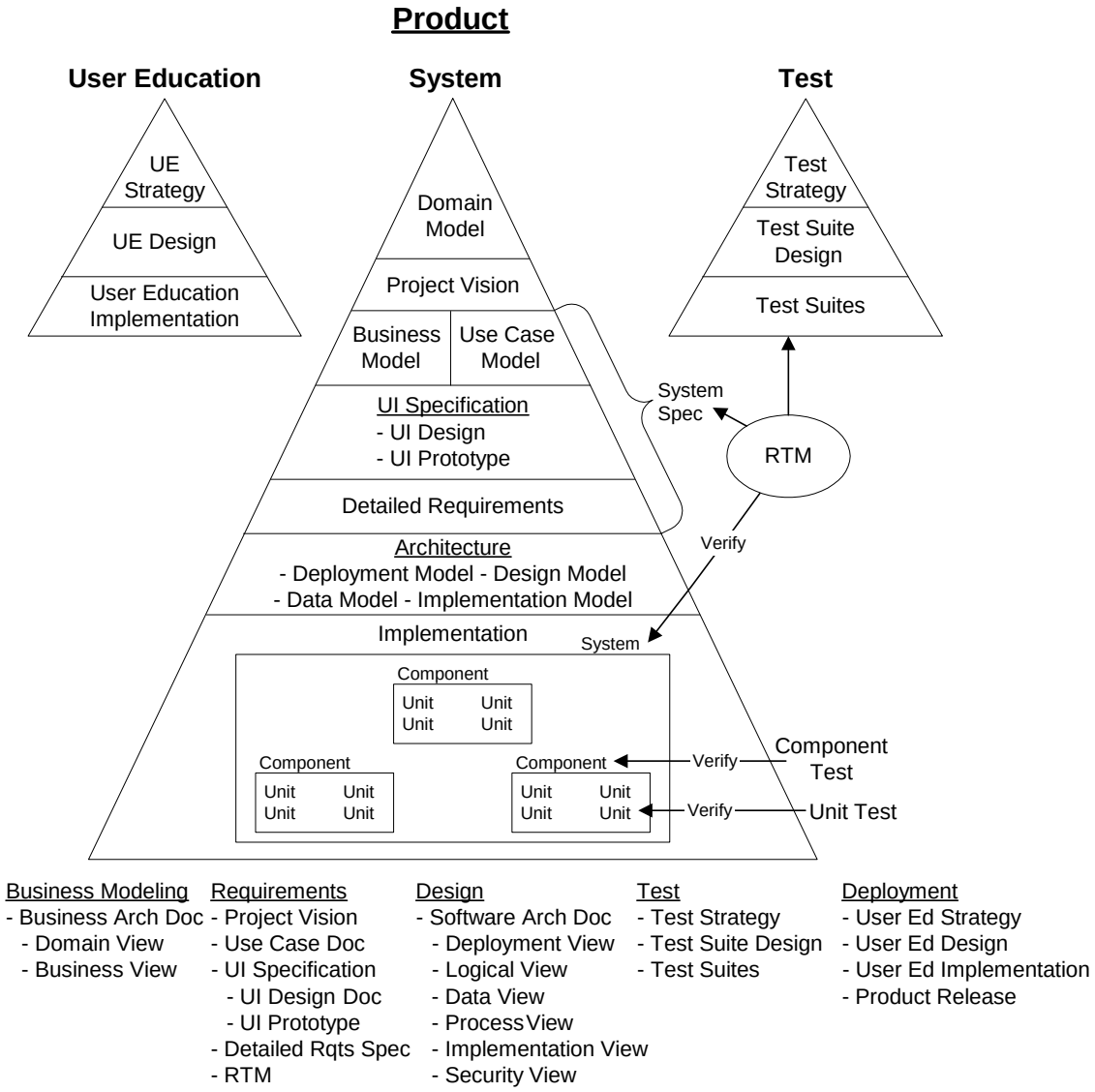
8.4 Non-repudiation & auditing

Omdat er vrijwel geen database bewerkingen door gebruikers worden uitgevoerd (met uitzondering van het invoeren, wijzigen en verwijderen van unit connectie gegevens) hoeft er geen auditing plaats te vinden.

BIJLAGE E.
TESTPLAN

INHOUDSOPGAVE

1 INLEIDING



Figuur 1-1 RUP producten en hun relaties

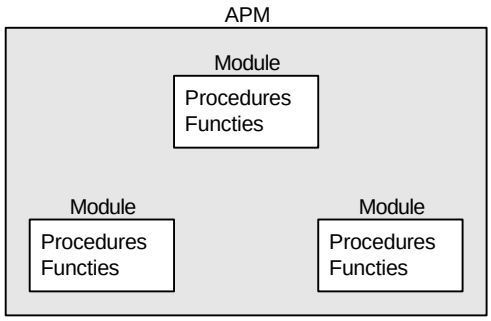
Het testplan beschrijft de testen die uitgevoerd zullen worden op het toekomstige systeem op drie manieren:

1. **Test Strategieën**: Definieert de diverse test sets;
2. **Test Set Ontwerpen**: Ontwerpt de diverse test sets;
3. **Test Sets**: Beschrijft de diverse tests.

2 TEST STRATEGIEËN

Het eerste stadium in het test process van het systeem is het definiëren van een test strategie. Een test strategie beschrijft de globale test wijze, identificeert de verschillende benodigde test niveaus en de methodes, technieken en gereedschappen die benodigd zijn. Het hart van de strategie is het analyseren van de eisen en het identificeren van de test sets, welke gezamenlijk controleren of het systeem aan alle systeem eisen voldoet. De strategie beschrijft ook de te gebruiken test data.

APM is modulair opgebouwd. Iedere module is ook weer opgebouwd uit procedures en functies.



Figuur 2-1 Opbouw van APM

Iedere bouwsteen van het systeem wordt apart getest voordat het geïntegreerd wordt met de andere bouwstenen. Dit met het resultaat dat de volgende soorten testen uitgevoerd zullen worden.

- **Procedure/functie test**: Controleer de werking van een procedure of functie
- **Module test**: Controleer de werking van een module
- **Integratie test**: Combineer twee modules en controleer de werking
- **Systeem test**: Controleer de werking van het gehele systeem.

2.1 System Test Strategy

Het proces voor het ontwikkelen van systeem testen zijn:

Stap 1: Analyseer alle systeem eisen;
Stap 2: Benader de systeem risico's vanuit een systeem test perspectief;
Stap 3: Definieer de test sets, gebruikmakend van de systeem risico's;
Stap 4: Leg de systeem eisen naast de test sets en sla dit op. Zorg dat de test sets bijgehouden worden wanneer het systeem evolueert;
Stap 5: Ontwerp de test cases per test set;
Stap 6: Creëer de test cases voor iedere test set.

2.2 Risico's, aannames en beperkingen

2.2.1 Risico's

ID	Omschrijving	Waar-schijn-lijkheid	Impact	Risico Factor
1	Veiligheidsrisico. Iemand kan proberen verbinding te maken met APM en playback data uitlezen. **Aanraden**: Testen in leesbaarheid en repliceerbaarheid van heen en weer verzonden data.	4	2	8
2	Veiligheidsrisico. Iemand kan proberen de database uit te lezen en unit connectie gegevens te achterhalen. **Aanraden**: Testen van DBMS beveiliging en isolatie.	5	6	30
3	Media Choice is een jong bedrijf met een relatief nieuwe dienst. Zenders en kabelmaatschappijen moeten vertrouwen krijgen om Media Choice toe te laten met hun signalen te werken. APM is een mogelijkheid om te bewijzen dat controle op playback uiterst scherp is. Wanneer het niet overtuigend is kan het vertrouwen niet ontstaan of beschadigd raken. **Aanraden**: Testen van duidelijkheid en uitgebreidheid van gegevens en alarmering bij problemen.	7	6	42
4	Het systeem kan uitvallen als gevolg van hardware storingen of software fouten. **Aanraden**: Testen op diverse vormen van externe storingen.	9	4	36
5	Het systeem zou playback fouten over het hoofd kunnen zien. **Aanraden**: Testen van playback gegevens in een bekende organisatie	5	8	40
6	Playback gegevens zouden verkeerd geïnterpreteerd kunnen worden door de controleur. **Aanraden**: Testen van duidelijkheid van foutrapportages.	5	8	40
7	Unit gegevens zouden verkeerd ingevoerd kunnen worden door de beheerder. **Aanraden**: Testen van invoer opties voor de beheerder	7	7	49
8	Het systeem zou na verloop van tijd alle bronnen van de gast computer opgemaakt kunnen hebben met vastlopers tot gevolg. **Aanraden**: Opzetten van duur testen	5	6	30

	met prestatiemeters om bronnen te meten.			

2.2.2 Aannames

Er zijn geen aannames bekend die de test strategie zouden kunnen beïnvloeden.

2.2.3 Beperkingen

Er zijn geen beperkingen waarmee rekening gehouden moet worden bij het opstellen van de test strategie.

2.2.4 Test Set Ontwerp Principes

De volgende principes zullen worden toegepast bij het ontwerpen van de test sets.

- Samenhang van testen binnen een test set moet zoveel mogelijk voorkomen worden.
- Samenhang van testen uit verschillende test sets mogen niet voorkomen. Iedere test set moet zelfstandig uitgevoerd kunnen worden.
- Test automatisering wordt gebruikt daar waar mogelijk.
- Elke test set begint met vooraf gedefinieerde data in de database.
- Testen moeten meerdere malen uitgevoerd kunnen worden zodat fouten gereproduceerd kunnen worden.

2.2.5 Systeem Test Sets

APM systeem testen bestaan uit drie test sets.

1. Core Functionaliteit/Belasting/Concurrency Test Set
2. Beveiliging/Beschikbaarheid Test Set
3. Gedetailleerde Test Set

2.2.6 Core Functionaliteit/Belasting/Concurrency Test Set (STS-1)

Dit test set verifieert dat APM het bedrijfskundige probleem wat het op moet lossen ook daadwerkelijk oplost. De units van Media Choice worden 24/7 gemoniteerd en problemen gedetecteerd en gerapporteerd.

Doelen

- Controleer dat het system het bedrijfskundige probleem zoals beschreven in de opdrachtomschrijving oplost.
- Controleer of APM kan werken met een hoeveelheid gegevens gelijk aan de hoeveelheid in een productie omgeving.
- Controleer de juistheid van de gegevens.

Ontwerp Benadering

- Probeer alle functies van het system uit. Invoeren, wijzigen en verwijderen van unit gegevens. Ophalen van data van de units. Analyseren van gegevens en rapporteren van resultaten.

- Laat APM functioneren met een grote hoeveelheid data (meer dan redelijkerwijs te verwachten in een productie omgeving). Vergroot daarna de hoeveelheid data totdat APM het niet meer aan kan.
- Controleer de juistheid van de databewerking door direct de database gegevens te benaderen.

Test Data Benadering
Creëer met de hand een kleine database met unit data. Vul de playout tabellen (schedulefiles en logfiles) met data uit het verleden van de units waarop fouten plaats vonden.

Test Automatisering Benadering
Bij het testen van het ophalen van playout gegevens zal gebruik gemaakt worden van een dummy unit. Een dummy unit is een software emulatie van een unit.

Uitvoer Benadering
Test cases zullen gemaakt worden tijdens de diverse project iteraties.

2.2.7 Beveiliging/Beschikbaarheid Test Set (STS-2)

Doelen

- Controleren of APM veilig is voor niet toegestane toegang.
- Controleren of APM robuust genoeg is voor de gewenste beschikbaarheid

Ontwerp Benadering

- Black box: Probeer van buitenaf in te breken op het systeem
- White box: Analyseren van de architectuur. Identificeer zwakke plekken. Probeer gebruik te maken van de zwakke plekken.

Test Data Benadering
Zorg dat de database in een vooraf gedefinieerde staat is en terug kan keren.

Test Automatisering Benadering
Automatisering van aanvallen zullen niet geautomatiseerd worden.

Uitvoer Benadering
Configureer het system zoveel mogelijk gelijk aan een APM system in een productie omgeving.

2.2.8 Gedetailleerde Test Set (STS-3)

Doelen

- Testen of alle gedefinieerde details ook daadwerkelijk in het systeem zitten.
- Uitvoerig testen van alle functies.

Ontwerp Benadering

- **GUI:** Toets ieder formulier aan de eisen.

- **Bedrijfskundige regels:** Controleer alle bedrijfskundige regels.

Test Data Benadering
Zorg dat de database in een vooraf gedefinieerde staat is en terug kan keren.

Test Automatisering Benadering
Automatisering van aanvallen zullen niet geautomatiseerd worden.

Uitvoer Benadering
Deze test set zal na alle andere test sets uitgevoerd worden.

2.2.9 Acceptatie Test (UAT)

Acceptatie van APM door Media Choice zal gebeuren wanneer alle systeem testen succesvol verlopen en er geen kritische fouten en ten hoogste twee, ontwijkbare, hoge prioriteit fouten bestaan. Test sets voor deze tests zullen samengesteld worden door een technische medewerker van Media Choice.

2.2.10 Systeem Verificatie Fase - Systeem Test Iteraties

Systeem Test Iteraties	**Tests**	**Geschatte Duur (uur)**	**Opmerkingen**
1	Core Functionaliteit / Belasting / Concurrency Test Set	2	Grote problemen in deze test set zullen waarschijnlijk de grootste wijzigingen in de software veroorzaken. Vandaar dat deze als eerst uitgevoerd worden.
2	Beveiliging / Beschikbaarheid Test Set	4	Deze set wordt als tweede uitgevoerd vanwege de grote waarde die aan veiligheid gehecht wordt.
3	Gedetailleerde Test Set	4	Falen in deze test zal waarschijnlijk de minste impact hebben op de software.
5	Regressie van alle test sets voor de acceptatie.	24	Sommige van de test sets zullen gelijktijdig uitgevoerd worden.

3 TEST SET ONTWERP

3.1 Core Functionaliteit/Belasting/Concurrency Test Set

ID	Test Case Name	Test Conditions
STS1-001	Invoeren unit	Voeg in een lege unit database 1 unit in zonder headend.
STS1-002	Invoeren headend en unit	Voeg in een lege database een headend in en koppel daar een nieuwe unit aan.
STS1-003	Invoeren headend en drie units	Voeg in een lege database een headend in en koppel daar drie nieuwe units aan.
STS1-004	Wijzigen gegevens	Wijzig achtereenvolgens in een headend gegevens en unit gegevens.
STS1-005	Bekijken fouten	In een database met foutmeldingen controleer of alle bekende foutmeldingen ook getoond worden.

3.2 Beveiliging/Beschikbaarheid Test Set

ID	Test Case Name	Test Conditions
STS2-001	Autorisatie	Controleer of de client uitsluitend werkt met 1 identificatiesleutel.
STS2-002	Data integriteit	Controleer of de data die opgehaald wordt ook daadwerkelijk zo op de unit opgeslagen is.
STS2-003	Data bescherming	Controleer of de database server uitsluitend benaderbaar is vanaf de APM server zodat aanvallen van buitenaf onmogelijk worden.
STS2-004	Beschikbaar-heid	Controleer het programma op memory leaks door het een hele nacht te laten draaien met minimaal 50 units in de database.

3.3 Gedetailleerde Test Set

ID	Test Case Name	Test Conditions
STS3-001	GUI eisen	Controleer of alle velden die ingevuld moeten kunnen worden uitsluitend relevante data toelaten. Controleer of er, wanneer een foutmelding geselecteerd wordt, de juiste details hiervan getoond worden.
STS3-002	Whitebox testen	Controleer wanneer unit/headend data ingevoerd wordt of deze altijd opgeslagen wordt. Controleer ook of dit niet gebeurd wanneer er geannuleerd wordt.