# *MPEG-21 Digital Item Adaptation (DIA)*

**Walter Brebels**

promotor :
Prof. dr. Wim LAMOTTE

Eindverhandeling voorgedragen tot het bekomen van de graad
master in de informatica multimedia

universiteit hasselt

# Dankwoord

Deze thesis is tot stand gekomen aan de Universiteit Hasselt dankzij de hulp van verschillende mensen. Graag zou ik deze hier dan ook even willen bedanken. Allereerst zou ik graag mijn promotor Prof. Dr. Wim LAMOTTE willen bedanken die mij de kans heeft gegeven om deze thesis tot stand te brengen. Vervolgens zou ik graag mijn begleider Dhr. Maarten WIJNANTS willen bedanken voor zijn hulp en advies bij de opbouw van deze thesis en voor het meermaals zorgvuldig nalezen van dit werk. Ik zou ook graag mijn medestudenten willen bedanken die mij vooral de onstpannende kant van Diepenbeek hebben laten zien. Tenslotte zou ik graag mijn ouders willen bedanken die mij de mogelijkheid en de middelen hebben gegeven om deze studierichting te volgen.

Hartelijk bedankt allemaal!

# Abstract

In deze thesis worden enkele nieuwe technologieën besproken die helpen bij het dynamisch adapteren van multimedia aan de noden van gebruikers. MPEG-21 heeft deze nieuwe werktuigen gestandaardiseerd in de hoop de heterogeniteit tussen de verschillende netwerken en apparaten te overbruggen bij het aanbieden van multimedia. De hele MPEG-21 standaard beschouwd alle componenten die betrokken zijn bij de creatie, distributie en consumptie van digitale media. In deze thesis worden de componenten die helpen bij de adaptatie van multimediabronnen nader bekeken.

Allereerst wordt een nieuw concept besproken dat geïntroduceerd is door MPEG-21. Het *Digital Item* is een container waarin multimedia en metadata op een hiërarchische manier gestructureerd worden. Dit concept brengt een uniforme structuur in de verschillende multimediabronnen waaruit een digitaal werkstuk kan bestaan. Het associeert bovendien het werkstuk met de werktuigen die vereist zijn bij de distributie en consumptie van het werkstuk. Deze werktuigen kunnen bijvoorbeeld zorgen voor de beveiliging van de inhoud van het werkstuk of de adaptatie van het werkstuk aan de noden van de gebruiker. De werktuigen zijn gestandaardiseerde XML-beschrijvingen die als metadata aan het Digital Item worden toegevoegd. De auteur van Digital Item kan hierdoor zelf specificeren hoe het Digital Item verwerkt moet worden bij de distributie en consumptie.

Vervolgens worden enkele werktuigen nader bekeken die helpen bij de adaptatie van multimediabronnen. De werktuigen bieden de mogelijkheid om de kwaliteit van schaleerbare multimediabronnen op een formaatonafhankelijke wijze te degraderen. Er wordt beschreven hoe we de werktuigen kunnen gebruiken om een optimale Quality of Service te garanderen voor de gebruiker. Deze werktuigen werken beiden onafhankelijk van de restricties die worden opgelegd door het systeem, het netwerk en de voorkeuren van de gebruiker.

Tenslotte wordt besproken hoe we deze werktuigen hebben geïmplementeerd om een adaptatie-engine te construeren. De adaptatie-engine laat gebruikers toe om op een transparante manier multimedia te ervaren die aan hun noden is aangepast.

# Inhoudsopgave

# Hoofdstuk 1

# Introductie

In de laatste jaren is het aanbod en verbruik van multimedia op het web sterk toegenomen. Niet alleen de hoeveelheid maar ook de diversiteit is toegenoemen. Digitale multimedia verschijnt in talloze vormen en de mogelijke formaten om deze te beschrijven blijven toenemen. Er bestaan al verschillende technologieën om infrastructuren mee samen te stellen die de distributie en de consumptie van multimedia mogelijk maken [Tim03]. De verschillende onderdelen van deze infrastructuren staan echter nog los van elkaar en het is niet altijd duidelijk hoe deze met elkaar kunnen interageren. In juni 2000 is het MPEG-21 standaardisatiecomité daarom voor het eerst begonnen met een nieuwe standaard te ontwikkelen die tracht een interoperabel raamwerk voor het transport en de consumptie van multimedia te creëeren [BPdWK06]. De visie voor de nieuwe standaard is [ISO04]:

> *"The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities."*

Ondertussen zijn er al acht jaar verstreken en zit de ontwikkeling van de standaard in zijn laatste stadia. MPEG-21 heeft in zijn multimediaraamwerk de componenten onderscheiden die aan bod komen tijdens de creatie, distributie en consumptie van digitale media. De hele standaard is een groot werk geworden dat opgedeeld is in verscheidene delen die grosso modo overeenkomen met de componenten van het multimediaraamwerk.

In de standaard wordt een nieuw centraal concept geïntroduceerd: een *Digital Item (DI)* [ISO05]. Dit is een container waarin multimediabronnen en metadata op een gestructureerde manier gekoppeld kunnen worden. De verschillende componenten die door MPEG-21 herkend zijn, werken in hun respectievelijke delen allemaal met de Digital Item container. De componenten beschrijven vaak tools die als metadata in het *eXtensible Markup Language (XML)* formaat aan bronnen gekoppeld kunnen worden in een DI. Eén van de componenten die door MPEG-21 herkend is, is het *Digital Item Adaptation (DIA)* gedeelte. In dit deel worden tools

voorzien die kunnen helpen bij het adapteren van multimediabronnen en metadata binnen een DI. Het is de kerncomponent binnen de MPEG-21 standaard waarmee de standaard het *Universal Multimedia Access (UMA)* begrip wil realiseren. Deze opvatting wil dat gebruikers multimedia overal, altijd en met elk apparaat kunnen consumeren.

Deze nieuwe standaard is niet de eerste internationale standaard ontwikkeld door MPEG. Het is de opvolger van verscheiden internationale standaarden die al bewezen hebben een groot succes te zijn. In 1988 is het MPEG comité begonnen met het ontwikkelen van MPEG-1, een standaard voor de codering van audio en video tot $1,5$Mbps [ISO91]. De standaard was ontwikkeld zodat analoge video en audio op digitale media kan opgeslaan worden; zoals bijvoorbeeld de toen zeer populaire Compact Disc (CD). De standaard gebruikt perceptuele encoderingsmethodes die aan de hand van de psychologische limitaties van de menselijke zintuigen data reduceren of zelfs weglaten [BPdWK06]. Het proces om audio en video te decoderen is gestandaardiseerd maar de encodering niet. Dit zorgde ervoor dat er verscheidene, al dan niet commerciële, implementaties zijn ontwikkeld die met elkaar concurreren. Het bekendste deel van deze standaard is ongetwijfeld het MPEG-1 Audio Layer 3 (MP3)-formaat dat nog steeds wereldwijd gebruikt wordt om audiobronnen te encoderen en op te slaan. De volgende standaard die het comité ontwikkelde was de MPEG-2 standaard. Deze standaard leverde betere resultaten voor hogere resoluties en bitrates [ISO94a]. Onder andere door de opkomst van de Digital Versatile Disc (DVD), een digitaal medium met een opslagcapaciteit die meer als zes keer zo groot dan die van een CD is, is deze standaard een groot succes geworden. De standaard wordt ook heel veel gebruikt voor digitale televisie. In 1998 is MPEG opnieuw begonnen met het ontwikkelen van een standaard, MPEG-4 [ISO94b]. Deze keer was het doel van de standaard om video met lage bitrates te coderen. Het resultaat bleek uiteindelijk een grote variëteit van bitrates aan te kunnen. MPEG-4 biedt ook verschillende nieuwe features zoals het mogelijk maken om het in combinatie met Digital Rights Management (DRM) software te gebruiken en interactiviteit. In 2001 heeft MPEG tenslotte zijn laatste standaard voor MPEG-21 gestandaardiseerd, MPEG-7 [ISO02]. Deze standaard onderscheidde zich van al de vorige standaarden doordat het geen standaard was om audio en/of video te coderen. Het is een XML-gebaseerde standaard om metadata te geven over multimediabronnen. Door metadata aan multimediabronnen te koppelen zouden we deze bijvoorbeeld doorzoekbaar kunnen maken.

## 1.1 De delen van de MPEG-21 standaard

De nieuwe MPEG-21 standaard kiest voor een heel andere aanpak als de vorige standaarden. Er werd nu meer naar het grote geheel gekeken. Al de componenten die betrokken zijn bij het creëeren, distribueren en consumeren van digitale media zijn apart behandeld

in verschillende delen. Samen vormen ze een groot multimediaraamwerk dat multimedia op een transparante manier naar de consument kan brengen. De verschillende delen waaruit de MPEG-21 standaard tot op heden is opgebouwd zijn [ISO04, BPdWK06]:

**Deel 1: Vision, Technologies and Strategy** geeft een visie en een doel voor de MPEG-21 standaard.

**Deel 2: Digital Item Declaration** (DID) geeft een standaard representatie voor een Digital Item. Er wordt een abstract model gedefinieerd met de entiteiten die nodig zijn om een Digital Item op te bouwen. Er wordt vervolgens een taal gestandaardiseerd om dit model in XML te beschrijven. Dit gedeelte wordt uitvoerig besproken in Hoofdstuk 2.

**Deel 3: Digital Item Identification and Description** legt uit hoe Digital Items op een unieke wijze geïdentificeerd kunnen worden. De relaties tussen bestaande identificatiestandaarden, zoals bijvoorbeeld ISBN-nummers, en identificatie van Digital Items wordt ook beschreven.

**Deel 4: Intellectual Property Management and Protection Components** geeft een implementatie van het DID model waarmee Digital Items op een geëncrypteerde manier kunnen beschreven worden. De bedoeling is om het multimediaraamwerk met bestaande *Digital Right Management (DRM)* technologieën te kunnen laten samenwerken.

**Deel 5: Rights Expression Language** biedt een nieuwe taal om de rechten op multimediabronnen te beschrijven. Met deze nieuwe taal kunnen we bijvoorbeeld auteursrechten op bronnen toepassen.

**Deel 6: Rights Data Dictionary** is een ontologie van termen die door de Rights Expression Language gebruikt kunnen worden om rechten uit te drukken.

**Deel 7: Digital Item Adaptation** beschrijft de tools om Digital Items op een transparante manier aan de omgeving van de gebruiker aan te passen. Er zijn tools gestandaardiseerd die: de omgeving van de gebruiker beschrijven; helpen bij adapteren van multimediabronnen en metadata; en *Quality of Service* van de adaptaties beheren. Dit gedeelte wordt gedetailleerd besproken in Hoofdstuk 3.

**Deel 8: Reference Software** is een stel van implementaties van de verschillende delen van de MPEG-21 standaard. De implementaties zorgen voor extra inzicht in de gestandaardiseerde tools en hun gebruik.

**Deel 9: File Format** voorziet de syntax voor een bestandsformaat waarmee we de declaratie van een DI en de bronnen waar het DI naar refereert in één bestand kunnen opslaan. Het bestandsformaat is gebaseerd op het ISO9660-formaat en functioneert als een package-formaat voor de verschillende bestanden waar een DI uit kan bestaan.

**Deel 10: Digital Item Processing** beschrijft een taal die Digital Items dynamisch kan maken door er functionaliteit aan toe te voegen. Met behulp van deze taal kunnen we Digital Items interactief maken.

**Deel 11: Evaluation Tools for Persistent Association** is een technisch rapport dat beschrijft hoe technieken geëvalueerd kunnen worden die een associatie leggen tussen metadata en multimediabronnen. Een techniek kan dit bijvoorbeeld doen door een watermerk toe te voegen aan de bron.

**Deel 12: Test Bed for MPEG-21 Resource Delivery** is een technisch rapport over een geïntegreerd MPEG-21 multimediaraamwerk. Het raamwerk kan gebruikt worden om nieuwe technologieën te testen die in samenwerking met MPEG-21 delen gebruikt kunnen worden.

**Deel 13:** was oorspronkelijk bedoeld voor Scalable Video Coding, dit is echter bij de MPEG-4 standaard toegevoegd. Deel 13 is nu niet ingevuld binnen de MPEG-21 standaard.

**Deel 14: Conformance** bevat een aantal bitstreams en XML-documenten waarmee getest kan worden als MPEG-21 implementaties voldoen aan de standaard.

**Deel 15: Event reporting** standaardiseert tools waarmee gebruikers het verbruik van multimediabronnen uit Digital Items kunnen traceren.

**Deel 16: Binary format** presenteert een manier om XML-documenten afhankelijk van hun schema zeer efficient binair te encoderen. XML-documenten bevatten veel redundantie, met deze encoderingsmethode kunnen we deze wegwerken en hierdoor veel bandbreedte besparen.

**Deel 17: Fragment Identification** specificeert een syntax om locaties of fragmenten in enkele bekende media formaten aan te duiden. Een string die aan deze syntax voldoet kan bijvoorbeeld als deel van een *Uniform Resource Locator (URL)* gebruikt worden. De syntax laat toe om locaties of fragmenten aan te duiden op basis van temporele, spatiale en logische eenheden.

**Deel 18: Digital Item Streaming** beschrijft een nieuwe taal waarmee Digital Items op streaming transportkanalen gemapt kunnen worden, zoals bijvoorbeeld RTP.

**Deel 19: Media Value Chain Ontology** is een voorstel voor een nieuw deel. Er wordt opgeroepen om een ontologie te creëeren die alle termen en hun relaties ten opzichte van elkaar definieert die voorkomen tijdens de creatie, distributie en consumptie van Digital Items. Deze ontologie zou gebruikt kunnen worden om DRM technologieën beter te kunnen integreren.

# Hoofdstuk 2

# Digital Item Declaration

Het hele MPEG-21 multimedia raamwerk is opgebouwd rond een nieuw centraal concept: een *Digital Item* [ISO05, BPdWK06, BDD05]. Digital Items worden binnen het raamwerk gebruikt om complexe digitale objecten voor te stellen. Het *Digital Item Declaration* is het tweede deel in de resem van de delen waaruit de MPEG-21 standaard bestaat. Het definieert een Digital Item als:

> *"a structured digital object with a standard representation, identification and metadata within the MPEG-21 framework".*

Het is eigenlijk een virtuele container waarin multimediabronnen en metadata op een hiërarchische manier zijn gestructureerd. Multimedia wordt tegenwoordig in steeds rijkere vormen gedistribueerd. Bijvoorbeeld bij een lied vervagen de grenzen al snel tussen het audiobestand met de audiodata, een tesktbestand met de songteksten of een afbeelding met de cover van het album. Met een Digital Item kunnen we deze als het ware aan mekaar plakken. Digital Items brengen structuur in de multimediabronnen en metadata die nodig is om een digitaal object te vormen zoals een tijdschrift, een CD of een fotoalbum. Binnen MPEG-21 zijn Digital Items ook de objecten waar de gebruiker mee interageert. Digital Items zijn de objecten die door gebruikers gecreëerd, geconsumeerd of gedistribueerd worden. Met het tweede deel van zijn standaard biedt het standaardisatiecomité van MPEG-21 een manier om zo een Digital Item op te bouwen en in het XML-formaat te declareren.

Het tweede gedeelte is onderverdeeld in drie delen [ISO05, BPdWK06, BDD05]. In het eerste deel wordt het *Digital Item Declaration Model (DID Model)* gedefinieerd. Dit is een stel van abstracte entiteiten die we als bouwstenen voor een Digital Item kunnen gebruiken. Er wordt hier geen syntax beschreven om het model in een specifieke taal te implementeren. Alleen de semantiek van de entiteiten en hun relaties ten opzichte van elkaar worden hier vastgelegd. Dit laat toe om het model in verschillende talen te implementeren of om mappings te leggen tussen al bestaande modellen en het MPEG-21 model voor een Digital Item. In het tweede deel wordt

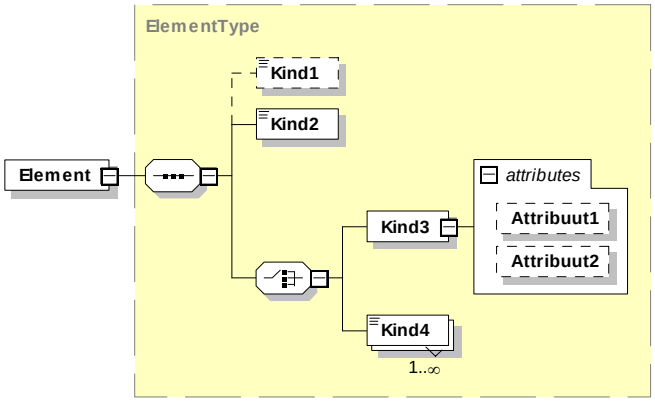
de syntax gestandaardiseerd om de bouwstenen van het model in XML te beschrijven. Dit wordt de *Digital Item Declaration Language (DIDL)* genoemd. De namen en de semantiek van de elementen die zijn toegelaten in deze nieuwe taal komen overeen met de bouwstenen van het model. De Digital Item Declaration Language voegt zelf nog een paar elementen en attributen toe die hulpzaam zijn bij het declareren van een Digital Item in XML. Het derde deel is een XML Schema dat de syntax van de Digital Item Declaration Language vastlegt.

## 2.1 Digital Item Declaration Language

Aangezien de elementen en hun semantiek in de Digital Item Declaration Language overeenkomen met de entiteiten in het achterliggende model, is het onnodig om deze twee keer te beschrijven. Daarom wordt hier alleen de syntax en de semantiek van de toegestane elementen in de Digital Item Declaration Language besproken. Als het model niet helemaal overeen komt met de syntax wordt dit specifiek vermeld. Tenzij anders vermeld komt de informatie in deze sectie uit [ISO05, BPdWK06, BDD05].

Voor elke elementdeclaratie die in deze sectie besproken wordt, zal een grafische representatie gegeven worden die de structuur van het element op een simpele en intuïtieve manier afbeeldt. De diagrammen zijn geconstrueerd met behulp van het software pakket XMLSpy, ontwikkeld door Altova [XMLb]. De diagrammen tonen aan welke afstammelingen een element kan bevatten en hoe deze erin mogen voorkomen. Als er relevante attributen toegelaten of verplicht zijn, worden deze eveneens getoond. In Figuur 2.1 wordt een fictief voorbeeld gegeven dat demonstreert hoe een datatype met zo een diagram afgebeeld wordt. De declaratie van het element `Element` wordt getoond met als datatype `ElementType`. De declaratie van een element is een kader met hierin de naam. De lijnen die vanuit deze kader vertrekken, beelden de boomstructuur van het element af. Lijnen die vanuit een elementdeclaratie vertrekken, monden uit in groeperingssymbolen. Deze symbolen groeperen enkele elementdeclaraties en/of andere groeperingssymbolen en bepalen hoe de kinderen hiervan mogen voorkomen in het element waardat het symbool een kind van is. Het eerste symbool dat we tegenkomen als we vanuit de declaratie van `Element` vertrekken is een sequentie. Dit symbool legt de volgorde vast van de kinderen die hieraan gekoppeld zijn. Vanuit het sequentiesymbool vertrekken drie lijnen. De eerste lijn mondt uit in een declaratie van het element `Kind1`. De kader van deze declaratie heeft een gestippelde lijn, dit betekent dat het element optioneel is. De volgende lijn mondt uit in een normale declaratie van het element `Kind2`. De laatste lijn leidt naar een nieuw groeperingssymbool, dit symbool is een keuze. Het symbool zegt dat één van de kinderen van het symbool mag voorkomen binnen de ouder van het symbool. `Kind3` of `Kind4` mag dus na `Kind2` voorkomen in `Element`. Aan `Kind3` kunnen ook twee attributen gegeven worden: `Attribuut1` en `Attribuut2`. De omkadering van de attributen heeft dezelfde be-

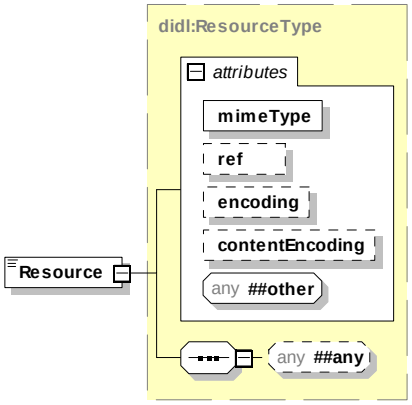
tekenis als bij elementen. Een gestippelde lijn betekent dat het optioneel is, een volle lijn betekent dat het verplicht is. In tegenstelling tot attributen kunnen elementen ook meer als één keer voorkomen binnen een andere element. `Kind4` toont aan hoe dit in het diagram afgebeeld wordt, de elementdeclaratie krijgt een dubbele kader met rechtsonder het minimum aantal keer dat het moet voorkomen, gevolgd door het maximum aantal keer dat het mag voorkomen. In `Element` mag na `Kind2` dus ofwel `Kind3` exact één keer voorkomen, ofwel `Kind4` minimum één keer.




Figuur 2.1: Grafische representatie van een XML Schema datastructuur door Altova XMLSpy

### 2.1.1 Resource

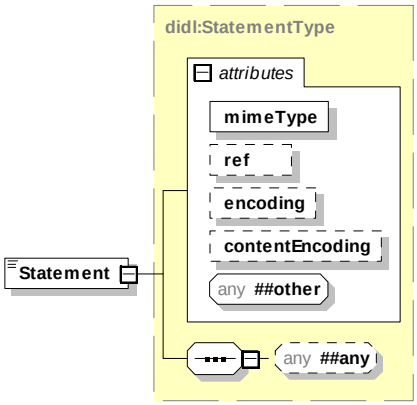
Resources zijn de multimediabronnen die bijdragen tot de vorming van het Digital Item. Een Resource kan bijvoorbeeld een afbeelding, een videofragment, een audiobron of een stuk tekst zijn. Een Resource kan zelfs geässocieerd worden met een fysiek object. Het *Resource*-element binnen de Digital Item Declaration Language kan naar een bron verwijzen of het kan de data van een digitale bron bevatten. We kunnen naar een bron verwijzen door middel van het *ref*-attribuut, dit kan een URI bevatten die een bron op een unieke wijze kan identificeren. Als we de binaire data van een bron in een Resource-element willen opnemen, moeten we deze eerst encoderen naar een tekstuele representatie aangezien we geen binaire data in een XML-document mogen plaatsen. De encodering die de bron is ondergaan moet meegegeven worden met het *encoding*-attribuut; de enige encodering die momenteel is toegelaten is de *base64*-encodering. Voordat we de binaire data naar een tekstuele representatie hebben omgezet, kan het zijn dat de bron nog andere encoderingmethodes is ondergaan om ze bijvoorbeeld te comprimeren. Deze lijst van encoderingmethodes moet meegegeven worden met het *contentEncoding*-attribuut. Met het *mimeType*-attribuut wordt het formaat van de originele bron aangegeven.




Figuur 2.2: Resource

### 2.1.2 Statement

Het *Statement*-element is syntactisch gezien identiek aan het Resource-element op de naam na, maar de semantiek ervan verschilt. Het kan dezelfde attributen hebben als een Resource-element die dezelfde betekenis hebben maar de betekenis van de inhoud van het element verschilt. Een Statement-element dient om onopgemaakte tekst of gestructureerde tekst aan een Digital Item toe te voegen. De tekst in een Statement is geen multimediale inhoud van het Digital Item, zoals tekst in een Resource-element, maar zal eerder functioneren als beschrijvende informatie, controle informatie of identificerende informatie. Een Statement-element kan bijvoorbeeld een MPEG-7 beschrijving bevatten of een beschrijving uit de Digital Item Adaptation-standaard. Een Statement-element is vaak gekoppeld aan een andere element in een Digital Item waardat de informatie op van toepassing is.

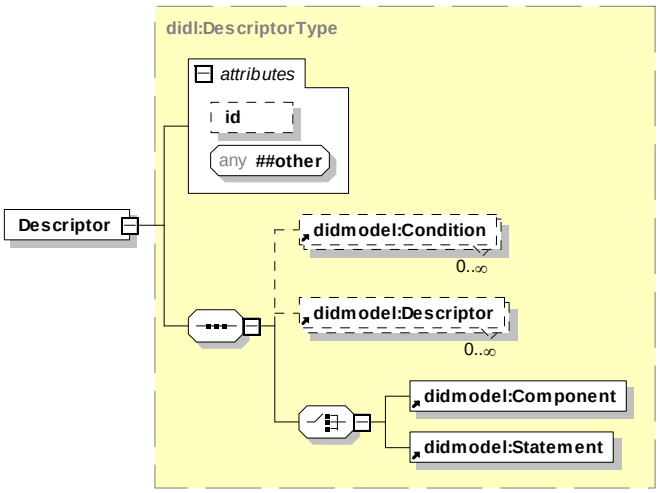



Figuur 2.3: Statement

### 2.1.3 Descriptor

Het *Descriptor*-element associeert informatie met het insluitende element. Deze informatie kan in de vorm van een Component of een Statement zijn. Een Component-element bevat, zoals in Sectie 2.1.4 nog verder uitgelegd zal worden, een multimediabron. Als we een Component in een Descriptor-element gebruiken, om dus als informatie met een ander element te associëren, dan kan dit bijvoorbeeld een miniatuurafbeelding zijn van de cover van een muziekalbum. Als we een Statement gebruiken zou dit bijvoorbeeld een MPEG-7 beschrijving van het album of de titel van het album kunnen zijn. In een Descriptor-element kunnen terug Descriptor-elementen worden toegevoegd die dan weer informatie geven over het huidige Descriptor-element.

De Descriptor-entiteit uit het DID Model heeft in de Digital Item Declaration Language twee syntactische vormen. De eerste is het Descriptor-element zoals zojuist beschreven. De tweede zijn attributen met een andere namespace. Elk element dat een Descriptor-element kan bevatten, mag dus ook attributen met een andere namespace bevatten.

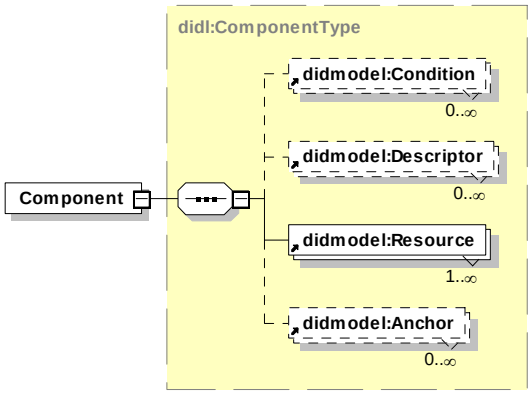



Figuur 2.4: Descriptor

### 2.1.4 Component

Een *Component*-element bindt Descriptor-elementen aan een multimediabron. De Descriptor-elementen dienen eerder om controle en structurele informatie te geven als beschrijvende of identificerende informatie. Een Component-element kan ook Anchor-elementen bevatten die speciale locaties of fragmenten binnen de bron aanduiden en hier informatie aan kunnen koppelen. De syntax en de semantiek van het Anchor-element wordt gedetailleerd besproken in Sectie 2.1.5.

In tegenstelling tot in het DID Model kan een Component-element in de Digital Item Declaration Language meerdere Resource-elementen bevatten. In het DID Model kan de component entiteit maar één resource entiteit bevatten. In de Digital Item Declaration Language horen meerdere Resource-elementen binnen één Component-element naar bitgewijs identieke multimediabronnen te verwijzen. Dit is gedaan zodat een Component-element een multimediabron kan beschrijven die op verschillende locaties beschikbaar is. De applicatie kan dan kiezen welke bron hij wilt gebruiken.



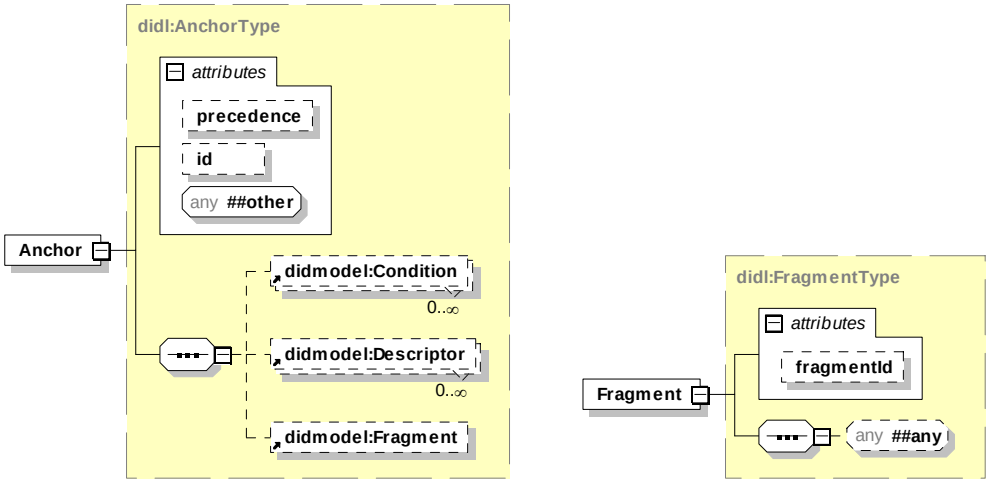

Figuur 2.5: Component

### 2.1.5 Anchor en fragment

Een *Fragment*-element bepaalt op een formaatafhankelijke wijze een specifieke locatie of een deel in een Resource. Dit kan gedaan worden door een *fragment identifier* [BLFM05] mee te geven in het fragmentId-attribuut en/of door metadata in het element te plaatsen. Als het fragmentId-attribuut is gespecificeerd en er is metadata toegevoegd dan is de metadata een verdere specificatie van de identifier in het fragmentId-attribuut. Een fragment identifier is afhankelijk van het formaat van de Resource waar het een locatie of een deel in moet bepalen. In deel 17 van de MPEG-21 standaard wordt de syntax voor fragment identifiers voor enkele bekende formaten gestandaardiseerd.

Een *Anchor*-element is een kind van een Component-element. Het bevat een Fragment-element dat een specifieke locatie of een deel in de multimediabron van deze Component aanduidt. Het Anchor-element koppelt Descriptor-elementen aan deze locatie of dit deel. Het kan een *precedence*-attribuut hebben dat de rangorde binnen de bovenliggende Component bepaalt. Door het id-attribuut van een Anchor kunnen we de multimediabron ook naar het Anchor laten verwijzen.

In Listing 2.1 wordt een Anchor-element getoond dat een stuk onopgemaakte tekst aan een




(a) Anchor (b) Fragment

Figuur 2.6: Anchor en fragment

```
<Component>
  <Resource mimeType="video/mp4" ref="http://someserver.com/video.mp4"/>
  <Anchor>
    <Descriptor>
      <Statement mimeType="text/plain">
       Introductie
      </Statement>
    </Descriptor>
    <Fragment fragmentId="mp(/~time('ntp', '0', '180')" />
  </Anchor>
</Component>
```

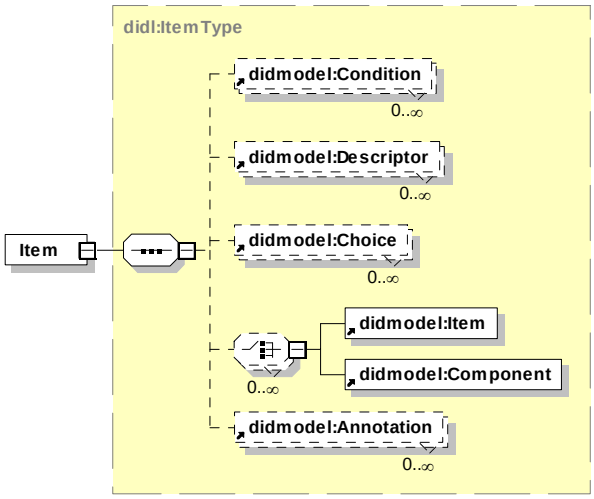
Listing 2.1: Een voorbeeld van het gebruik van een Anchor-element

deel van een videobron koppelt. Het fragmentId-attribuut bevat een fragment identifier die voldoet aan deel 17 van de MPEG-21 standaard. Deze fragment identifier specificeert de eerste drie minuten van de videobron. Met het Anchor-element koppelen we de text '`Introductie`' eraan.

### 2.1.6 Item

Een *Item*-element stelt een logisch onverdeelbare entiteit voor of een compilatie hiervan. Een Item kan bijvoorbeeld één lied of een muziekalbum, een compilatie van liederen, zijn. Een Item-element kan bestaan uit meerdere sub-Items en/of Componenten. Als het nog sub-Items bevat wordt het beschouwd als een compilatie, anders als een logisch onverdeelbare entiteit. De Component-elementen zijn de multimediabronnen die we aan het Item koppelen, het zijn de bouwstenen die een Item vormen. Een Item kan ook Descriptor-elementen bevatten, deze

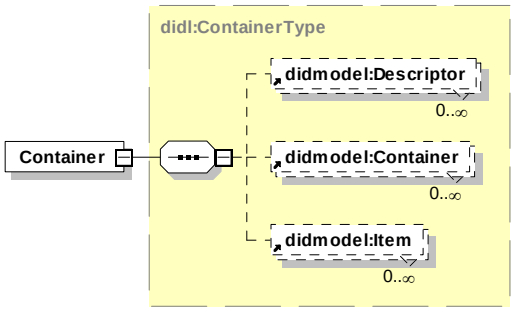
zullen vooral beschrijvende en identificerende informatie geven over het Item.




Figuur 2.7: Item

### 2.1.7 Container

Het *Container*-element dient om groeperingen van Item(s) en/of Container(s) voor te stellen. Aan een Container kunnen ook Descriptor-elementen gebonden worden. Het element kan vergeleken worden met een schap, die een collectie boeken of CD's kan bevatten.




Figuur 2.8: Container

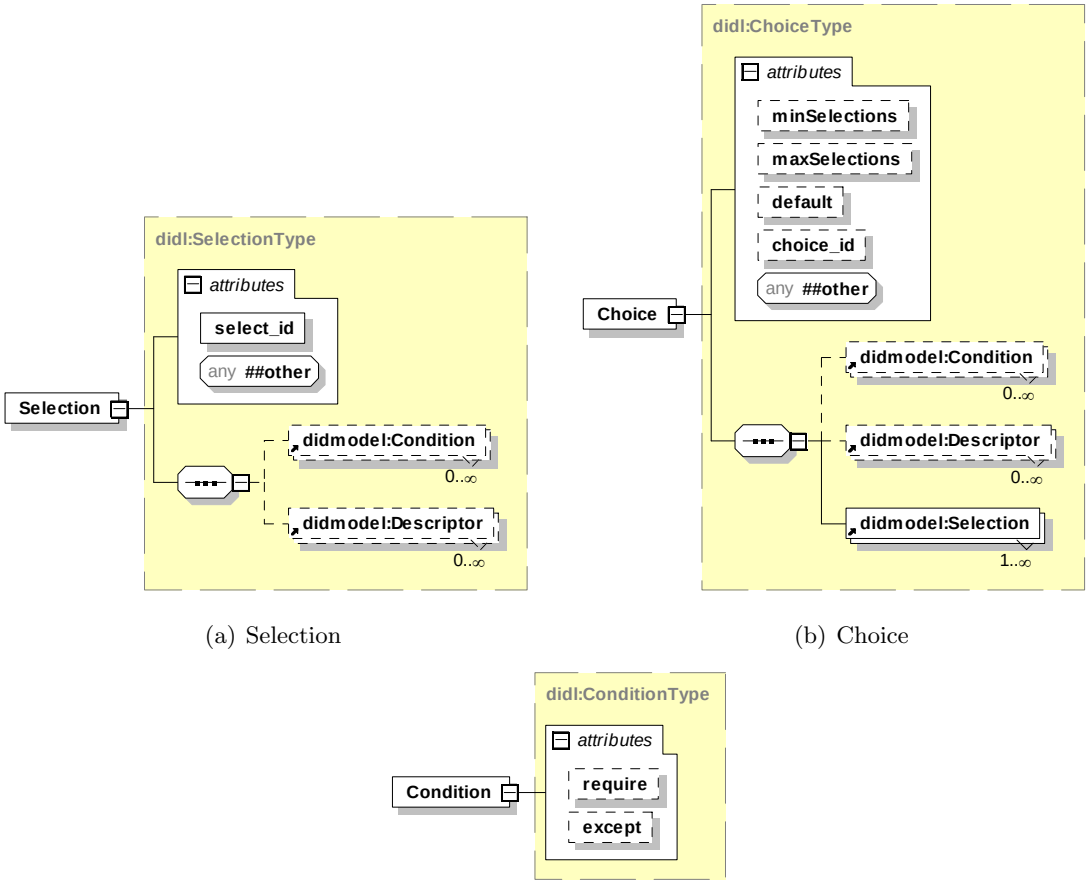
### 2.1.8 Selection, choice en condition

Al de tot nu toe besproken elementen dienen om een Digital Item op een hiërarchische wijze te structureren. Het DID Model en de Digital Item Declaration Language voorzien ook respectievelijk de entiteiten en de elementen die het mogelijk maken om Digital Items *at run-time* te configureren. De overeenkomst tussen de elementen en de entiteiten is echter wel niet zo triviaal als bij de andere elementen.

Het DID Model specificeert vier entiteiten die het configureren mogelijk maken: *predicate*, *selection*, *choice* en *condition*. Een predicate is een expressie die evalueert naar waar, onwaar of onbeslist. Een choice is een reeks van selections waarvan de applicatie kan kiezen om ze te selecteren of niet, dit proces configureert het Digital Item. Een selection is geässocieerd met een predicate. Als de applicatie een selection selecteert in een choice dan is het geässocieerde predicate waar; als de applicatie het niet selecteert is het geässocieerde predicate onwaar; en anders is het geässocieerde predicate onbeslist. Verschillende entiteiten uit het model kunnen condition-entiteiten bevatten, dit betekent dat ze voorwaardelijk kunnen gemaakt worden. Een condition-entiteit zegt dat de entiteit waaraan het gekoppeld is, afhankelijk is van een aantal predicaten en/of de ontkenning van een ander aantal predicaten. Met andere woorden, een condition-entiteit kan een aantal predicaten en/of de ontkenning van een ander aantal predicaten bevatten en zal naar waar evalueren als al deze predicaten naar waar evalueren en de ontkennende predicaten naar onwaar evalueren. Een entiteit dat gekoppeld is aan condition-entiteiten zal alleen door de applicatie beschouwd worden als één van zijn condition-entiteiten naar waar evalueert. De selection- en choice-entiteiten kunnen zelf ook voorwaardelijk gemaakt worden waardoor er een complexe beslissings-structuur gecreëerd kan worden.

In de Digital Item Declaration Language zijn er gelijknamige elementen voor een selection, choice en een condition. Een predicate wordt voorgesteld door een stringwaarde. Een *Selection*-element heeft een *select_id*-attribuut waarmee het predicate kan meegegeven worden waar het aan gekoppeld is. Een *Choice*-element heeft de extra attributen *minSelections* en *maxSelections* waarmee aangegeven kan worden hoeveel selecties er mogen of moeten gemaakt worden. Het *default*-attribuut van een Choice-element bepaalt de standaard selectie. Met het *choice_id*-attribuut kunnen we een Choice-element op een unieke manier binnen een Digital Item identificeren. Het *Condition*-element heeft twee attributen. Het *require*-attribuut is een lijst met predicaten die naar waar moeten evalueren en het *except*-attribuut is een lijst met predicaten die naar onwaar moeten evalueren eerdat het Condition-element naar waar zal evalueren.

In Listing 2.2 wordt een voorbeeld gegeven van een Item-element uit een Digital Item dat een lied beschrijft. Het Item-element heeft een Descriptor-element dat beschrijvende informatie over het lied verschaft, in dit geval de auteur en de titel in onopgemaakte tekst. Er zijn twee Component-elementen opgenomen in het Item die beiden naar een multimediabron met het lied verwijzen, maar de bronnen hebben een verschillend formaat. Elk Component-element is voorwaardelijk en hangt af van een predicaat. Het Item-element bevat een Choice-element dat ons dwingt om één van deze predicaten te kiezen. In het Choice-element bevinden zich twee Selection-elementen. Afhankelijk van welk we selecteren wordt één Component voor de




(a) Selection

(b) Choice

(c) Condition

Figuur 2.9: Selection, choice en condition

applicatie beschikbaar en de andere niet. Aan zowel het Choice-element als aan de Selection-elementen is beschrijvende informatie gekoppeld. Deze zouden we aan een menselijke gebruiker kunnen presenteren als we de keuze aan hem of haar overlaten. We zouden hier ook informatie in een gecontroleerd formaat in kunnen opnemen zodat een applicatie die bewust is van de decodeermogelijkheden van het systeem automatisch een keuze kan maken.

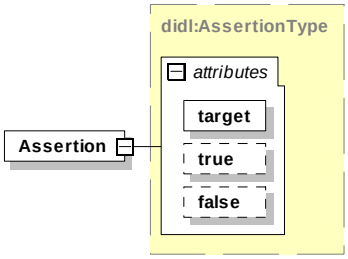
### 2.1.9 Assertion

Met *Assertion*-elementen kunnen predicaten op waar of onwaar ingesteld worden. Assertion-elementen kunnen zo gebruikt worden om de (gedeeltelijke) staat van een een Choice-element te bewaren. Een Assertion-element heeft drie mogelijke attributen. Het *target*-attribuut is een URI naar het Choice-element waar het naar refereert. Het *true*-attribuut bepaalt welke predicaten uit dit Choice-element waar zijn en het *false*-attribuut bepaalt welke predicaten uit dit Choice-element onwaar zijn. Alle andere predicaten uit het Choice-element zullen

```
<Item>
  <Descriptor>
    <Statement mimeType="text/plain">
      John Lennon - Imagine
    </Statement>
  </Descriptor>
  <Choice minSelections="1" maxSelections="1" default="MP3_FORMAT">
    <Descriptor>
      <Statement mimeType="text/plain">
        Welk audio formaat zou u willen gebruiken?
      </Statement>
    </Descriptor>
    <Selection select_id="WMA_FORMAT">
      <Descriptor>
        <Statement mimeType="text/plain">
          Windows Media Audio
        </Statement>
      </Descriptor>
    </Selection>
    <Selection select_id="MP3_FORMAT">
      <Descriptor>
        <Statement mimeType="text/plain">
          MPEG-1 Audio Layer 3
        </Statement>
      </Descriptor>
    </Selection>
  </Choice>
  <Component>
    <Condition require="WMA_FORMAT" />
    <Resource mimeType="audio/x-ms-wma"
      ref="http://someserver.com/Imagine.wma" />
  </Component>
  <Component>
    <Condition require="MP3_FORMAT" />
    <Resource mimeType="audio/mpeg"
      ref="http://someserver.com/Imagine.mpg3" />
  </Component>
</Item>
```

Listing 2.2: Een configureerbaar Item-element uit een Digital Item

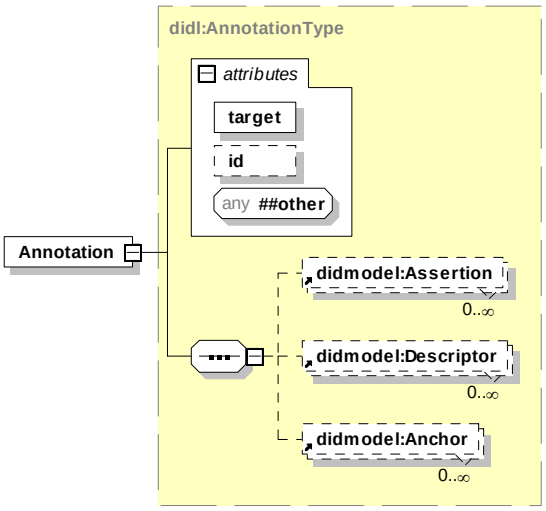
onbeslist zijn. De applicatie kan dan nog beslissen om overige selections, die geässocieerd zijn met een predicaat dat hier niet in voorkomt, te selecteren of niet.




Figuur 2.10: Assertion

### 2.1.10 Annotation

*Annotation*-elementen kunnen gebruikt worden om informatie aan andere elementen toe te voegen, zonder deze elementen te wijzigen. Met het *target*-attribuut kunnen we een URI meegeven die een element binnen het huidige Digital Item of binnen een ander Digital Item identificeert. De elementen in het Annotation-element zijn dan gekoppeld aan dit element. In het Annotation-element kunnen we drie elementtypes hebben: Assertion-elementen, waarmee we bijvoorbeeld de configuratie van een Item-element kunnen bewaren; Descriptor-elementen, waarmee we bijvoorbeeld zelf commentaar aan een element kunnen toevoegen; of Anchor-elementen die we bijvoorbeeld kunnen gebruiken om informatie over specifieke locaties binnen een bron toe te voegen.




Figuur 2.11: Annotation

### 2.1.11 DIDL, DIDLInfo en Declarations

Al de voorgaande elementen hebben een gelijknamige entiteit binnen het DID Model met dezelfde semantiek. De Digital Item Declaration Language voegt zelf ook nog drie elementen toe die alleen van toepassing zijn bij het beschrijven van een Digital Item in XML, ze zijn dus geen onderdeel van de opbouw van een Digital Item.

Het *Declarations*-element laat toe om enkele van de hierboven besproken elementen te declareren zonder ze te instantiëren binnen het Digital Item. Als we deze elementen in een Digital Item willen instantiëren dan kunnen we dit doen met behulp van de *W3C XML Inclusions (XInclude)*-standaard [Con06]. Deze standaard laat toe om een XML-document of een specifiek deel hiervan in een XML-document te voegen. Met behulp van deze standaard refereren we in een Digital Item dus naar een element in het Declarations-element. Als er nergens in het Digital Item naar een element uit het Declarations-element wordt gerefereerd, dan is dit dus ook geen deel van het Digital Item. Door elementen in het Declarations-gedeelte te declareren kunnen we sommige elementen één keer declareren en daarna gemakkelijk herbruiken, zoals bijvoorbeeld een copyright-statement. De elementen die in een Declarations-element zijn toegelaten zijn: Item, Descriptor, Component, Annotation of Anchor.

Het *DIDLInfo*-element kunnen we gebruiken om applicatie-specifieke informatie in het document op te nemen die van toepassing is op het document zelf maar niet op het Digital Item dat het document beschrijft.

Het *DIDL*-element is het wortelelement van een Digital Item declaratie in de DIDL-taal. Het kan een DIDLInfo-element bevatten dat eventueel gevolgd wordt door een Declarations-element. Het moet of een Container- of een Item-element bevatten dat het Digital Item zal representeren.

## 2.2 Een Digital Item

In Listing 2.3 wordt een uitgewerkt voorbeeld van een Digital Item in de Digital Item Declaration Language getoond. Het Digital Item beschrijft een configureerbaar digitaal fotoalbum. Er wordt gedemonstreerd hoe dat het album aangevuld kan worden met MPEG-7 metadata. Er wordt ook getoond hoe dat gebruikers informatie, zoals commentaar, kunnen toevoegen zonder dat deze echt deel uitmaakt van het album. Door het Digital Item te configureren kunnen we dan beslissen als we deze informatie willen zien of niet. Er wordt ook getoond hoe dat de configuratie van het Digital Item bewaard kan worden.

Het bovenste *Item*-element dat we tegenkomen representeert het album. Het eerste kind hiervan is een *Choice*-element dat ons toelaat het album te configureren. Er zijn twee *Selecti-*

*on*-elementen, elk geassocieerd met een predicaat: `TOON_TAGS` en `TOON_AUTEUR`. Als het eerste predicaat geselecteerd is, wordt metadata beschikbaar die door de gebruiker is toegevoegd. Als het tweede predicaat geselecteerd is wordt metadata over de auteur van het album beschikbaar. We zouden deze keuze aan de gebruiker kunnen presenteren; op de achtergrond door de applicatie zelf laten uitvoeren; of zoals zodra nog verder uitgelegd zal worden door een *Assertion*-element laten invullen. Het volgende element dat we tegenkomen is een *Descriptor*. Dit element bevat de metadata die met het album is geässocieerd. Hierin wordt de naam van de auteur van het album in een MPEG-7 beschrijving gegeven. We zouden hier ook gerust meer informatie in kunnen plaatsen zoals waar en wanneer het album is gecreëerd. In het element hangt een *Condition*-element dat zegt dat de informatie alleen beschikbaar moet worden gesteld als het predicaat `TOON_AUTEUR` in de bovenstaande keuze is gekozen.

Het volgende element dat we tegenkomen is een *Item*-element dat een foto uit het album representeert. Het element bevat een *Component*-element met een verwijzing naar de afbeelding in het JPEG2000 bestandsformaat. Er zijn twee *Resource*-elementen gegeven, deze moeten naar bitgewijs identieke bronnen verwijzen. In dit geval is de afbeelding op twee servers beschikbaar voor als er één niet beschikbaar zou zijn. In het *Item*-element vinden we ook een *Annotation*-element terug. Dit voegt informatie toe aan een onderdeel van het Digital Item, zonder dat deze informatie echt deel uitmaakt van dit onderdeel. In dit voorbeeld heeft een gebruiker een tag op een foto geplaatst, hiermee bedoelen we dat een gebruiker een regio in de afbeelding heeft geselecteerd en hier commentaar op heeft gegeven. Het *Annotation*-element verwijst naar het *Component*-element binnen het Digital Item, dit betekent dat de elementen in de annotatie hieraan worden toegevoegd. Het *Anchor*-element bevat de commentaar die is toegevoegd en een *Fragment*-element met hierin een MPEG-7 beschrijving die de regio binnen de afbeelding bepaalt. De gebruiker heeft in de afbeelding een vierkant geselecteerd van positie $(100, 100)$ tot positie $(200, 200)$, en hier de tekst "Jos" aan gekoppeld.

Aan het hoofd *Item*-element dat het album representeert, is tenslotte ook nog een *Annotation*-element gekoppeld dat informatie aan dit *Item*-element toevoegt. De informatie hierin is weeral geen deel van de logische entiteit "Album" dat het *Item* representeert. De informatie die toegevoegd wordt is een *Assertion*-element dat zegt dat het `TOON_TAGS` predicaat waar is en het `TOON_AUTEUR` predicaat onwaar is. Deze annotatie wordt hier gebruikt om de configuratiestaat van het album op te slaan.

```
<DIDL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Item id="album0">
    <Choice minSelections="0" maxSelection="2" choice_id="choice0">
      <Selection select_id="TOON_TAGS" />
      <Selection select_id="TOON_AUTEUR" />
    </Choice>
    <Descriptor>
      <Condition require="TOON_AUTEUR" />
      <Statement mimeType="text/xml">
        <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001">
          <Description xsi:type="CreationDescriptionType">
            <CreationInformation>
              <Creation>
                <Title>Vakantie juli 2008</Title>
                <Creator>
                  <Role href="urn:mpeg:mpeg7:cs:RoleCS:2001:AUTHOR"/>
                  <Agent xsi:type="PersonType">
                    <Name>
                      <GivenName>Jan</GivenName>
                      <FamilyName>Doedel</FamilyName>
                    </Name>
                  </Agent>
                </Creator>
              <Creation>
            </CreationInformation>
          </Description>
        </Mpeg7>
      </Statement>
    </Descriptor>
    <Item>
      <Component id="image1">
        <Resource mimeType="image/jp2"
          href="http://server.albums.com/image1.jp2" />
        <Resource mimeType="image/jp2"
          href="http://server2.albums.com/image1.jp2" />
      </Component>
      <Annotation target="image1">
        <Anchor>
          <Condition require="SHOW_TAGS" />
          <Descriptor>
            <Statement mimeType="text/xml">
              Jos
            </Statement>
          </Descriptor>
          <Fragment>
            <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001">
              <Description xsi:type="ContentEntityType">
                <MultimediaContent xsi:type="ImageType">
                  <Image>
                    <StillRegion>
                      <SpatialMask>
                        <SubRegion>
```

```
                        <Box mpeg7:dim="2 2">100 100 200 200</Box>
                      </SubRegion>
                    </SpatialMask>
                  </StillRegion>
                </Image>
              </MultimediaContent>
            </Description>
          </Mpeg7>
        </Fragment>
      </Anchor>
    </Annotation>
  </Item>
  <!-- ... meerdere foto's ... -->
  <Annotation target="album0">
    <Assertion target="choice0" true="TOON_TAGS" false="TOON_AUTEUR" />
  </Annotation>
 </Item>
</DIDL>
```

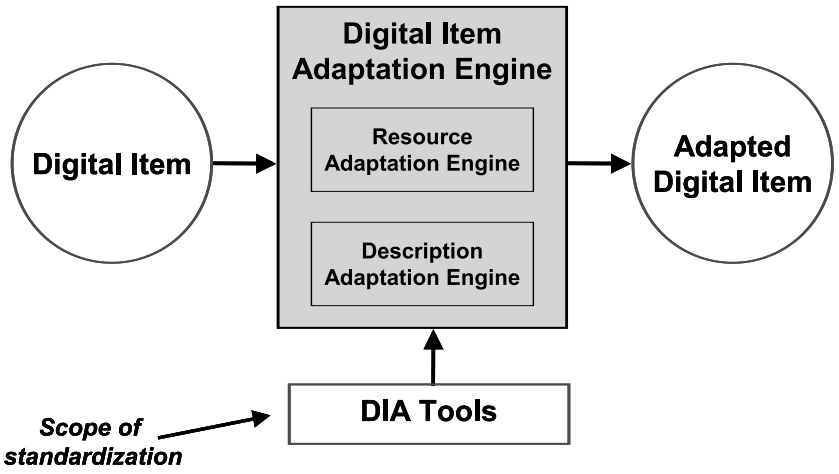
Listing 2.3: Digital Item voor een fotoalbum

# Hoofdstuk 3

# Digital Item Adaptation

In de laatste jaren is het aanbod en verbruik van multimedia op het web sterk toegenomen. Mede door het feit dat gebruikers tegenwoordig probleemloos zelf multimediale bronnen kunnen creëeren, cirkelt het vandaag op het web rond met tal van audio- en video-fragmenten. Multimedia op het web wordt steeds gevarieerder en komt in steeds meer verschillende formaten voor. Gebruikers die de multimedia consumeren beschikken over een grote diversiteit aan apparaten die met dynamische en heterogene netwerken verbonden zijn. De specificaties en de mogelijkheden van de apparaten kunnen sterk verschillen. Gebruikers willen toegang hebben tot multimedia op apparaten zoals computers, set-top boxen, Personal Digital Assistants of GSMs. De netwerken waarover deze apparaten de media ontvangen of versturen verschillen ook zeer sterk. Apparaten kunnen met netwerken zoals ethernet, Wi-Fi of GPRS verbinden. Hierbovenop heeft elke gebruiker ook nog specifieke voorkeuren om de multimedia af te spelen. Deze grote verscheidenheid van apparaten en netwerken zorgt voor sterk uiteenlopende restricties op de transmissie en het consumeren van multimedia. Het standaardisatie-comité van MPEG-21 wil de kloof tussen de heterogeniteit van apparaten en netwerken en het grote aanbod van multimedia in talrijke formaten overbruggen [Vet04, VT05]. De MPEG-21 standaard streeft ernaar om multimedia op een transparante en interoperabele wijze aan te bieden [ISO03]. De transparantie zorgt ervoor dat gebruikers geen last van de systeem- en netwerk-restricties ondervinden. Interoperabiliteit betekent dat verschillende applicaties de multimediale inhoud kunnen uitwisselen en hier mee te werk kunnen gaan. Er wordt getracht om *Universal Multimedia Access (UMA)* aan te bieden [Vet04, VT05]. Dit houdt in dat multimedia overal, altijd en op elk apparaat aangeboden en verbruikt kan worden.

De MPEG-21 *Digital Item Adaptation (DIA)* standaard voorziet de middelen om Digital Item Adaptation-engines te bouwen [ISO03]. Een adaptatie-engine kan de kwaliteit van Digital Items adapteren aan de wensen van de gebruiker en de restricties van zijn omgeving. Een engine kan onderverdeeld worden in twee delen: een *resource adaptation engine* die de




Figuur 3.1: MPEG-21 DIA concept [ISO03]

binaire bronnen binnen een Digital Item aanpast; en een *description adaptation engine* die de metadata binnen een Digital Item aanpast. MPEG-21 standaardiseert geen adaptatie-engine, maar wel de *tools* die een engine kunnen helpen bij het automatisch selecteren en adapteren van een Digital Item [ISO03]. In Figuur 3.1 is de rol van de DIA-standaard binnen een engine afgebeeld. Deze tools zijn XML-beschrijvingen die we kunnen toevoegen als metadata aan een Digital Item. MPEG-21 standaardiseert de syntax en de semantiek hiervan. Met de tools die de DIA standaard biedt kunnen we adaptatie-engines maken die onafhankelijk zijn van het systeem waar ze voor aanpassen en van het formaat van binaire bronnen in een Digital Item.

De tools die in de standaard zijn beschreven, zijn [ISO03, BPdWK06]:

- **Usage Environment Description**
  Dankzij de Usage Environment Description tools kunnen we onze adaptatie-engines onafhankelijk maken van de gebruiker waar we voor adapteren. Binnen de MPEG-21 standaard zijn gebruikers niet alleen eindgebruikers, maar iedereen die aan het proces deelneemt om een Digital Item te creëeren, distribueren en consumeren. We moeten daarom niet alleen weten wat een gebruiker kan verbruiken, maar ook wat hij kan maken of doorsturen. De Usage Environment Description tools zijn opgedeeld in vier categorieën die de mogelijkheden en de restricties van de gebruiker beschrijven [VT05]:
  - De *User Characteristics* beschrijven gebruikerspecifieke informatie. Dit kunnen persoonlijke gegevens zijn zoals een adres, een naam of een geschiedenis van vorige interacties waar een gebruiker bij betrokken was. Hierin kunnen ook voorkeuren voor het presenteren van Digital Items of informatie over de toegankelijkheids

karakteristieken van de gebruiker voorkomen. Deze karakteristieken kunnen bijvoorbeeld een handicap van de gebruiker beschrijven.

  - De *Terminal Capabilities* beschrijven de mogelijkheden van het systeem van de gebruiker. Dit houdt in: de formaten waar het apparaat naar kan encoderen en/of decoderen; de aanwezig invoer- en uitvoer-apparaten en hun eigenschappen, zoals bijvoorbeeld als er een scherm aanwezig is en welke resolutie dit aankan; en tenslotte eigenschappen van het apparaat zelf, zoals bijvoorbeeld het stroomverbruik of de CPU-kracht.
  - De *Network Characteristics* beschrijven de statische en dynamische eigenschappen van het netwerk. Statische eigenschappen kunnen bijvoorbeeld de maximum en de minimum gegarandeerde bandbreedte zijn. Een dynamische eigenschap kan bijvoorbeeld het gemiddeld aantal pakketen dat per seconde verloren gaat, zijn.
  - De *Natural Environment Characteristics* beschrijven de omgeving van de gebruiker. Hiermee kunnen we bijvoorbeeld de locatie en tijd van de gebruiker aangeven of eigenschappen zoals de belichting of het niveau van geluidshinder rond de gebruiker.

- **Bitstream Syntax Description**
  De Bitstream Syntax Description tools zijn ontwikkeld om binaire bronnen in een schaleerbaar formaat te adapteren. Bronnen in een schaleerbaar formaat zijn opgebouwd in lagen zodat we de kwaliteit van de bron kunnen degraderen door bepaalde lagen weg te laten. Er is een nieuwe taal ontwikkeld waarmee de syntax van een binair formaat beschreven kan worden. Met een beschrijving van een formaat in deze taal kunnen we een Bitstream Syntax Description van een bitstream genereren. Een Bitstream Syntax Description is een XML-document dat de bitstream op een hiërarchische wijze beschrijft. De bron wordt aangepast door de beschrijving te transformeren en deze transformatie te reflecteren op de bitstream. Doordat we de syntax van nieuwe binaire formaten kunnen beschrijven en we met schaleerbare bronnen werken kunnen we een formaatonafhankelijke module maken die bronnen efficient kan adapteren. We hebben dus geen aparte software meer nodig voor elk formaat. De Bitstream Syntax Description tools worden uitvoerig behandelt in Sectie 3.1.

- **Terminal and Network Quality of Service**
  De Terminal and Network Quality of Service tools helpen een adaptatie-engine bij het automatisch selecteren van optimale adaptatieparameters, zodat de aangepaste versie van de bron aan de restricties opgelegd door het systeem en het netwerk van de gebruiker voldoet. Deze tool koppelt daarom restricties aan adaptatieoperaties en de resulterende kwaliteit van deze adaptatie. Een gedetailleerde uitwerking van deze tools is te vinden in Sectie 3.2.

- **Bitstream Syntax Description Link**
  De Bitstream Syntax Description Link tools laten toe om adaptatie-architecturen te ontwerpen door een link te leggen tussen de Terminal and Network Quality of Service tools en de BSD tools. De BSDLink tools kunnen de gekozen adaptatieparameters mappen op de input van het proces dat de BSD van een bitstream transformeert. Deze tools worden gedetaileerd besproken in Sectie 3.3.

- **Metadata Adaptability**
  De Metadata Adaptability tools geven informatie over de metadata binnen een Digital Item. In een Digital Item kan gemakkelijk metadata met multimedia bronnen gekoppeld worden. De bedoeling van deze metadata is om tekstuele informatie over bijvoorbeeld een audio- of video-bron te geven. Hierdoor kunnen we er bijvoorbeeld voor zorgen dat een audio- of video-bron doorzoekbaar is zoals een tekst. Bij grote bronnen of zeer gedetaileerde metadata kan deze informatie al snel enorme proporties aannemen. Sommige apparaten hebben niet de benodigde geheugenruimte of processortijd om deze metadata te verwerken of soms is de bandbreedte van het netwerk ook niet voldoende om de hele metadata mee over te sturen. Als een bron aangepast wordt kan het ook zijn dat niet alle metadata nog relevant is voor de gebruiker. Als we metadata gaan aanpassen door er de metadata uit te halen die relevant is voor de gebruiker noemen we dit filteren, als we de complexiteit van de metadata willen reduceren dan noemen we dit schalen. De metadata adaptability tools kunnen een metadata adaptatie-engine hierbij helpen door informatie te geven over de metadata zoals het totaal aantal elementen, de grootte van de metadata in bytes of informatie over specifieke elementen in de metadata zoals hun maximum diepte of de waarden die ze kunnen aannemen. De Metadata Adaptability tools voorzien eigenlijk metadata over metadata binnen een Digital Item.

  Soms zijn er verschillende metadata beschrijvingen beschikbaar voor één bron en dan moeten deze geïntegreerd worden. Er kunnen twee problemen optreden als we verschillende beschrijvingen voor één bron hebben. Delen van de metadata uit de verschillende beschrijvingen kunnen dubbel voorkomen en deze moeten dan geïntegreerd worden, of delen kunnen elkaar aanvullen en dan moeten deze samengevoegd worden. Dankzij de metadata over deze metadata, die door de tools voorzien kan worden, kunnen metadata adaptatie-engines belangrijke gegevens over de originele metadata te weten komen zonder eerst de hele metadata te hoeven analyseren.

- **Session Mobility**
  De Session Mobility tools zijn ontwikkeld om de staat van een Digital Item tussen gebruikers over te brengen of eventueel te bewaren. De staat van een Digital Item bestaat uit de configuratiestaat, dit zijn de waarden van de predicaten die in de choice-

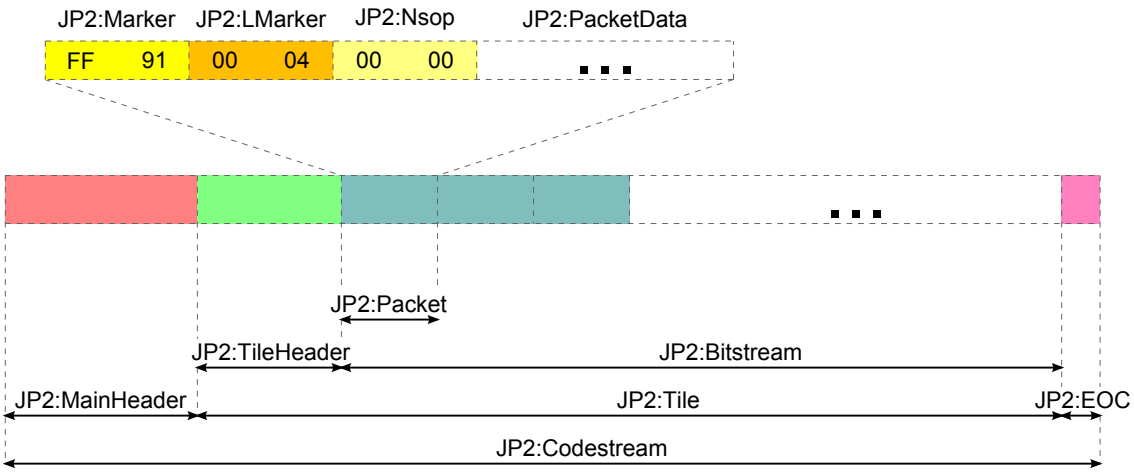
elementen zijn bepaald, en eventuele applicatie-specifieke informatie. Deze applicatie-specifieke informatie kan bijvoorbeeld zijn welk lied uit een muziekalbum nu aan het afspelen is. De Session Mobility regels stellen daarom regels op om een nieuwe Digital Item, een Context Digital Item genaamd, te creëeren waar deze staat in is bevat. Het Context Digital Item wordt gekoppeld aan het originele Digital Item door binnen het Context Digital Item te refereren naar het originele Digital Item.

- **DIA Configuration**
  Met de DIA Configuration tools kan de maker van een Digital Item specificeren waar hij of zij wil dat de adaptatie gebeurt. In een DIA Configuration beschrijving kunnen we aanduiden welke DIA-beschrijvingen door de verbruiker, de aanbieder of éénder wie behandeld moeten worden. Als de DIA-beschrijvingen voor de adaptatie bijvoorbeeld door de aanbieder behandeld moeten worden, gaat de aanbieder de adaptatie uitvoeren en de beschrijving niet meer doorsturen naar de verbruiker. Als de DIA-beschrijvingen voor adaptatie door de verbruiker behandeld moeten worden, gaat de aanbieder ze wel doorsturen en zelf geen adaptatie uitvoeren. In het andere geval kan de aanbieder kiezen wat hij doet. We kunnen ook specificeren welke Choice-elementen uit het Digital Item op de achtergrond geconfigureerd moeten worden en welke door de gebruiker geconfigureerd moeten worden.

## 3.1 Bitstream Syntax Description

De *Bitstream Syntax Description Tools (BSD Tools)* zijn ontwikkeld om multimediabronnen op een formaatonafhankelijke wijze te adapteren [ISO03, BPdWK06]. Ze werken met bronnen in een schaleerbaar formaat. Dit betekent dat de bron gedegradeerd kan worden door bepaalde datasegmenten uit de bitstream weg te laten. Er kunnen dus verschillende kwaliteits-versies van een bron bekomen worden door simpele operaties op de bitstream uit te voeren zoals datatruncatie in plaats van complexe, CPU-intensieve conversies. Er zijn echter wel nog veel verschillende formaten voor de multimediabronnen en er is een generieke methode nodig om deze te adapteren.

De BSD Tools steunen op het feit dat we in het gecomprimeerde domein van de bron werken en de bron kunnen adapteren door louter datasegmenten weg te laten en kleine bitstream wijzigingen door te voeren, zonder dat we eerst de bron moeten decoderen. Om dit op een generieke wijze te doen wordt het concept van een *Bitstream Syntax Description (BSD)* geïntroduceerd [ISO03]. Een BSD is een document gestructureerd in het *eXtensible Markup Language (XML)*-formaat dat de structuur van de bitstream op een hiërarchische wijze beschrijft. In een BSD heeft elk element een binaire representatie. Elementen kunnen




Figuur 3.2: Visuele representatie van de BSD uit Listing 3.1

hierdoor overeenkomen met datastructuren of logische groeperingen van datastructuren uit de bitstream, zoals bijvoorbeeld een frame in een videofragment. Het is belangrijk dat de schaleerbaarheid van de bitstream gereflecteerd wordt in de BSD en dat de BSD zelf ook schaleerbaar is. De bitstream wordt dan geadapteerd door met de BSD te werken en deze te transformeren naar een andere, geadapteerde, BSD. Sommige elementen in de BSD bevatten een tekstuele representatie van de binaire data waardoor we de waarde kunnen aanpassen tijdens een transformatie, andere elementen verwijzen gewoon naar een datasegment in de originele bitstream waardoor we het segment alleen maar kunnen weglaten tijdens een transformatie en niet wijzigen. De hiërarchie van de elementen in de BSD beschrijft de structuur van de binaire data in de bitstream. Omdat we in een XML-document de volgorde van attributen niet kunnen vastleggen maar de structuur van elementen wel, is het belangrijk dat binaire data alleen maar in elementen en niet in attributen bevat mag zijn. In principe kunnen we alle data van de bitstream in de elementen opnemen maar dat zou de BSD onnodig groot maken. Daarom is de BSD een aanvullend document dat beschouwd kan worden als metadata van de bitstream.

In Listing 3.1 wordt een voorbeeld van een BSD gegeven die een JPEG2000-bitstream [ISO00] beschrijft. De hele uitwerking van de boomstructuur gevormd door de elementen wordt niet getoond wegens plaatsgebrek. Alleen de meest bovenliggende elementen en de uitwerking van één tak wordt hier ter illustratie gegeven. In Figuur 3.2 wordt een visualisatie van de hiërarchie van de elementen gegeven. Het `Codestream`-element is het root-element en omvat de hele bitstream. Deze is onderverdeeld in drie delen: het `Mainheader`-, het `Tile`- en het `EOC`-element. Het `MainHeader`-element is de header van de afbeelding; hierin staan velden met informatie over de spatiale resolutie, het aantal gebruikte kleurlagen en gebruikte encoderingstechnieken. Het middenste element is het `Tile`-element, dit beschrijft de geëncodeerde data van de afbeel-

```
<DIA>
  <Description xsi:type="BSDType">
    <Codestream xmlns="JP2"
                xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
                xmlns:jp2="JP2">
      <MainHeader>
      <!-- MainHeader elements -->
      </MainHeader>
      <Tile>
        <TileHeader>
        <!-- TileHeader elements -->
        </TileHeader>
        <Bitstream>
          <Packet>
            <SOP>
              <Marker>FF91</Marker>
              <LMarker>4</LMarker>
              <Nsop>0</Nsop>
            </SOP>
            <PacketData>155 435</PacketData>
          </Packet>
          <!-- ... more packets ... -->
        </Bitstream>
      </Tile>
      <EOC>
      <!-- End Of Codestream marker -->
      </EOC>
    </Codestream>
  </Description>
</DIA>
```

Listing 3.1: Deel uit de BSD van een JPEG2000-bestand.

ding. Een afbeelding die geëncodeerd wordt met het JPEG2000-algoritme kan eerst opgedeeld worden in een aantal tegels die dan ieder afzonderlijk worden geëncodeerd. In dit geval is de hele afbeelding geëncodeerd als één tegel. Een tegel bevat op zijn beurt ook weer een header, het `TileHeader`-element. De geëncodeerde data van een tegel is opgedeeld in afzonderlijke pakketten. Het zijn deze pakketten die we kunnen weglaten om een gedegradeerde versie van de afbeelding te bekomen. Een `Packet`-element bestaat uit een `Marker`-, `LMarker`-, `Nsop`- en een `PacketData`-element. Deze elementen bevatten geen andere elementen meer en hebben dus een rechtstreekse binaire representatie. Het `Marker`-element geeft aan dat de komende blok data een pakket is. Het bestaat uit twee bytes en wordt in de BSD voorgesteld als een hexadecimale string. Het `LMarker`-veld bepaalt de lengte van de header van het blok data zonder het `Marker`-veld, in dit geval dus het aantal bytes dat het `LMarker`-veld en het `Nsop`-veld samen innemen. Het `Nsop`-element is een unsignedShort dat een volgordenummer voor pakketten is. Het `PacketData`-element verwijst naar de geëncodeerde data van dit pakket in de originele bitstream. De eerste waarde is de startpositie in de originele bitstream en de tweede waarde is de lengte. Het laatste element van de bitstream is het `EOC`-element, dit bevat enkele bytes die het einde van de bitstream markeren.

De procedure om een bitstream te adapteren met behulp van de BSD Tools bestaat nu uit drie fases [BPdWK06]:

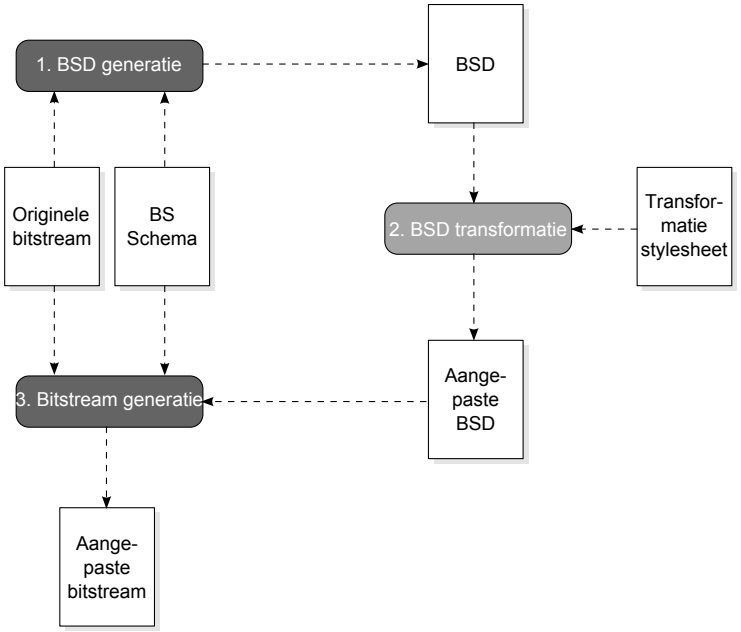
1. De genereratie van de BSD van de bitstream, ook wel het *BintoBSD*-proces genoemd in de DIA standaard;

2. De transformatie van de BSD, dit proces is niet gestandaardiseerd maar er worden wel bestaande technologieën gesuggereerd; en

3. De generatie van een nieuwe geadapteerde bitstream op basis van de aangepaste BSD, ookwel het *BSDtoBin*-proces genoemd.

Voor de eerste en derde fase van de adaptatieprocedure zijn er twee tools ontwikkeld. De eerste is de *Bitstream Syntax Description Language (BSDL)* en kan gebruikt worden voor beide processen. Het is een taal om schema's in te beschrijven die de syntax van een bitstream definiëren. De taal beschrijft de relatie tussen de elementen in een BSD en hun binaire representatie in de bitstream. De schema's worden *Bitstream Syntax Schema's (BS Schema's)* genoemd en kunnen gebruikt worden om een bitstream van een BSD te genereren en, afhankelijk van het schema, ook om een BSD van een bitstream te genereren. De hele procedure om een bitstream te adapteren met deze tool is afgebeeld in Figuur 3.3. Het nadeel van deze schema's is dat het schema altijd nodig is tijdens de twee fases om de relatie tussen de elementen en de binaire representatie aan te duiden en dat voor elke formaat een eigen schema nodig is. De tweede tool, het *generic Bitstream Syntax Schema (gBS Schema)* probeert een

oplossing te bieden voor enkele van deze problemen. Het gBS Schema is een gestandaardiseerd schema in de BSDL-taal. Het schema bevat enkele abstracte datatypes waar zoveel mogelijk bitstreams op een optimale manier mee beschreven kunnen worden. Doordat de relatie tussen de elementen van een BSD die aan dit schema voldoet en hun binaire representatie gestandaardiseerd is, is het schema niet meer nodig tijdens de generatie van een nieuwe bitstream (in dit geval ook wel het *gBSDtoBin*-proces genoemd). Maar door de abstractheid van de datatypes en het generieke karakter van het schema kan het wel niet gebruikt worden om een BSD te genereren. Beide tools zullen in de rest van deze sectie in meer detail besproken worden.




Figuur 3.3: Het proces om een bitstream te adapteren met behulp van de BSD Tools.

De tweede fase van de adaptatieprocedure, de transformatie van de BSD, is niet gestandaardiseerd. Er bestaan wel al verscheidene technologieën om XML-documenten te transformeren. Eén van die techonlogieën is de *eXtensible Stylesheet Language Transformations (XSLT)*-standaard [Kay07]. XSLT-stylesheets kunnen aan de hand van een aantal parameters XML-documenten transformeren in andere XML-documenten. XSLT-stylesheets maken hevig gebruik van XPath expressies om elementen in het brondocument te localiseren. Dit vereist dat een XSLT-processor het brondocument eerst helemaal inleest en hier een boomrepresentatie van opbouwt alvorens de transformatie uitgevoerd kan worden. Hierdoor is deze technologie niet geschikt voor een streamingscenario of voor applicaties die maar beschikking hebben tot een beperkte geheugengrootte. Een alternatieve methode zijn de *Streaming Transformations for XML (STX)*-stylesheets [CBN+07]. STX-stylesheets werken op events

van het brondocument, zoals het starten en eindigen van elementen, en hebben hierdoor geen last van de nadelen van XSLT-stylesheets.

### 3.1.1 Bitstream Syntax Description Language

Met de Bitstream Syntax Description Language kunnen we Bitstream Syntax Schema's ontwikkelen [ISO03]. BS Schema's zijn schema's in een generiek formaat dat de binaire structuur van een multimediaformaat beschrijft. Een generiek proces kan dan van een bitstream en een BS Schema dat het formaat van de bitstream beschrijft een BSD genereren en vice versa. MPEG-21 standaardiseert zelf geen schema's voor specifieke formaten dus een BSD voor een bitstream is niet uniek bepaald.

De BSDL is gebaseerd op de W3C XML Schema standaard [Con04]. De XML Schema standaard is een formaat om de structuur van XML-documenten te beschrijven. Schema's in dit formaat beschrijven waar en hoe elementen in een XML-document mogen voorkomen en welke waarden ze mogen bevatten. Met een schema in dit formaat kunnen dus eigenlijk gestructureerde datatypes worden beschreven. Het primaire doel van zulke schema's is de validatie van een XML-document. De BSDL voegt hier een nieuwe functionaliteit aan toe, namelijk het genereren van een BSD uit een bitstream en vice versa. Om BS Schema's hiervoor geschikt te maken, bevat de BSDL een stel van uitbreidingen en restricties op de XML Schema standaard.

#### Datatypes in de Bitstream Syntax Description Language

Met XML Schema kunnen eigen datatypes beschreven worden. Er wordt een onderscheid gemaakt tussen complexe en simpele datatypes. Een complex datatype is een gestructureerd datatype dat andere complexe en/of simpele datatypes bevat. Een simpel datatype in XML Schema bestaat uit drie aspecten: de *value space*, de *lexical space* en de *facets*. De value space zijn alle mogelijke waarden welke het datatype kan aannemen. De lecical space zijn de tekstuele representaties van deze waarden, en de facets zijn restricties op de waarden die het datatype kan aannemen. De BSDL voert een vierde aspect toe aan het simpele datatype: de binaire representatie. De BSDL laat alleen maar simpele datatypes van de XML Schema standaard toe waarvoor er een impliciete binaire representatie is.

Een voorbeeld van een datatype dat niet is toegelaten is het *xsd:integer*-datatype. Dit datatype kan alle mogelijke gehele waarden bevatten; aangezien de waarden niet begrensd zijn is er dus ook geen impliciete binaire representatie voor. Het *xsd:int*-datatype daarentegen is afgeleid van het *xsd:integer*-datatype maar beperkt de mogelijke waarden van -2147483648 tot 2147483647. Hierdoor kan het datatype in 4 bytes worden voorgesteld. Alle datatypes van de XML Schema standaard die wel zijn toegelaten zijn afgebeeld in Tabel 3.1 samen met

| Datatype | Aantal bytes |
|---|---|
| `string` | onbepaald |
| `float` | 4 |
| `double` | 8 |
| `hexBinary` | onbepaald |
| `base64Binary` | onbepaald |
| `long` | 8 |
| `int` | 4 |
| `short` | 2 |
| `byte` | 1 |
| `unsignedLong` | 8 |
| `unsignedInt` | 4 |
| `unsignedShort` | 2 |
| `unsignedByte` | 1 |

Tabel 3.1: Toegelaten XML Schema datatypes in de BSDL.

hun binaire lengte in bytes. Niet alle datatypes hebben een vaste lengte. Voor datatypes zonder vaste lengte zoals bijvoorbeeld een string zijn er speciale constructies nodig waarmee de lengte bepaald kan worden. Deze zullen verderop nog besproken worden.

**Restricties in de Bitstream Syntax Description Language**

In een XML Schema kunnen nieuwe simpele datatypes gecreëerd worden door ze van andere simpele datatypes af te leiden en hier facets aan toe te voegen. Facets zijn restricties op de mogelijke waarden van het datatype. Ze hebben echter geen invloed op de binaire representatie ervan en worden daarom genegeerd door BSDL parsers.

Hier is één uitzondering op en dat is het *maxExclusive*-facet. Als het maxExclusive-facet bij een unsigned integer type (*unsignedByte*, *unsignedShort*, *unsignedInt*, of *unsignedLong*) wordt gebruikt, wordt hiermee de bitgrootte van het datatype bepaald. De bitgrootte van het datatype is dan het logaritme met grondtal twee van de waarde van het facet, afgerond naar de dichtsbijzijnde hogere gehele waarde. Als het maxExclusive-facet bij een verdere afleiding van het datatype nog eens wordt gebruikt, wordt dit genegeerd.

In Listing 3.2 wordt weergegeven hoe een nieuw datatype met een specifiek aantal bits wordt gedefinieerd. Het datatype is afgeleid van het *xsd:unsignedShort*-datatype door middel van een *xsd:restriction*. Met een *xsd:restriction* nemen we een basistype, in dit geval een *xsd:unsignedShort*, en creëeren we een nieuw datatype door een facet toe te voegen dat de waarden die het basistype kan aannmen, beperkt. Hier voegen we het *maxExclusive*-facet toe,

```
<xsd:simpleType name="a4BitInteger">
  <xsd:restriction base="xsd:unsignedShort">
    <xsd:maxExclusive value="16"/>
  </xsd:restriction>
</xsd:simpleType>
```

Listing 3.2: Een datatype definite van een 4-bit unsigned integer waarde door middel van het maxExclusive-facet.

dat de waarden verder beperkt tot waarden tussen 0 en 16. Hierdoor kunnen alle mogelijke waarden van het nieuwe datatype met 4 bits worden voorgesteld.

#### Uitbreidingen in de Bitstream Syntax Description Language

De BSDL bevat een stel van uitbreidingen op de huidige XML Schema standaard. De XML Schema standaard is ontworpen om uitbreidbaar te zijn en laat dit op twee manieren toe. Een eerste manier bestaat eruit om *annotation*-elementen toe te voegen aan elementen in een schema. Annotation-elementen kunnen op hun beurt *appinfo*-elementen bevatten waar applicatiespecifieke (XML-)data in geplaatst kan worden. De tweede manier bestaat eruit om attributen met een andere namespace als die van XML Schema toe te voegen aan elementen. Zowel de annotation-elementen als de attributen worden genegeerd door een XML Schema parser. Hierdoor kan een BS Schema nog altijd gebruikt worden om de syntax van een BSD te valideren. De BSDL laat toe om BS Schema's op dezelfde manier uit te breiden. Elementen in *appinfo*-elementen en attributen met een niet-BSDL namespace worden genegeerd door BSDL parsers.

De uitbreidingen en restricties van de BSDL zijn in twee categorieën verdeeld, afhankelijk van het proces waar ze voor nodig zijn. Elk stel van uitbreidingen is in een apart schema ondergebracht dat geïmporteerd dient te worden in het BS Schema dat ze gebruikt. De twee categorieën zijn:

- De BSDL-1 uitbreidingen bevatten een attribuut en enkele nieuwe datatypes. Ze worden gebruikt bij het genereren van een bitstream van een BSD. Ze zijn vastgelegd in een schema met als namespace *'urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS'*.

- De BSDL-2 uitbreidingen zijn nodig zodat we van een BS Schema en een bitstream een BSD kunnen genereren. Een standaard XML Schema is hier niet voor gemaakt en de BSDL-2 constructies lossen de problemen op die we hierbij tegenkomen. Het schema importeert het BSDL-1 schema dus BS Schema's die deze uitbreidingen gebruiken dienen het BSDL-1 schema niet nog eens te importeren. De definities zijn ondergebracht in een schema met als namespace *'urn:mpeg:mpeg21:2003:01-DIA-BSDL2-NS'*.

### De BSDL-1 uitbreidingen en restricties

De BSDL-1 uitbreidingen bestaan uit één attribuut en twee semantisch gelijkwaardige datatypes:

- **ignore**
  Het ignore-attribuut kan een boolean waarde bevatten en kan aan elementen in een BSD worden toegevoegd. Elementen met de waarde van het ignore-attribuut op true worden genegeeerd tijdens het omzetten van een BSD naar een bitstream. Dit laat toe om elementen met een semantische waarde aan een BSD toe te voegen die niet in de resulterende bitstream worden opgenomen. Er kunnen zo elementen aan een BSD toegevoegd worden die door een specifieke applicatie gebruikt worden, maar door een generieke BSDL parser genegeerd worden.

- **byteRange**
  Het byteRange-datatype is een lijst van twee positieve integer waarden die een binair segment in de bitstream voorstellen. De eerste en de tweede waarde verwijzen naar respectievelijk de startpositie en de lengte van het segment.

- **bitstreamSegment**
  Het bitstreamSegment-datatype is semantisch gelijk aan het byteRange-datatype. De syntax is echter anders. Bij dit datatype worden de startpositie en de lengte meegegeven door attributen. Het datatype is alleen aan het BSDL-1 schema toegevoegd voor compatibiliteit met het gBS Schema. Dankzij dit en het byteRange-datatype kunnen we verwijzen naar binaire data in de bitstream, en moet dus niet alle binaire data in de BSD worden inbegrepen.

### De BSDL-2 uitbreidingen en restricties

De BSDL-2 uitbreidingen en restricties zijn nodig in een BS Schema als we met dit schema en bijhorende bitstreams BSDs willen genereren. De BSDL-2 uitbreidingen en restricties kunnen in twee categorieën opgedeeld worden: de structurele uitbreidingen en de uitbreidingen die van toepassing zijn op datatypes.

De structurele uitbreidingen zijn:

- **rootElement**
  Het rootElement-attribuut wordt aan het wortelelement van het BS Schema meegegeven en bepaalt met welke top-level element declaratie in het schema we onze BSD moeten beginnen. Een XML Schema document kan namelijk verschillende top-level element declaraties hebben, maar de BSD mag maar één wortelelement hebben.

- **nOccurs**

  Datatypes of delen van kunnen in een bitstream typisch meerdere keren na mekaar voorkomen, bijvoorbeeld frames in een videobestand. Met het nOccurs-attribuut kunnen we bepalen hoeveel keren sommige delen zich herhalen. Het nOccurs-attribuut kan meegegeven worden aan een XML Schema particle. Een XML Schema particle is een elementdeclaratie of een groepering van elementdeclaraties. In XML Schema kunnen we op drie manieren elementdeclaraties groeperen: een *xsd:sequence* zegt dat de elementen in de volgorde zoals ze hierin gedeclareerd zijn, moeten voorkomen; een *xsd:choice* zegt dat exact één van de elementen moet voorkomen; en een *xsd:all* zegt dat al de elementen die hierin gedeclareerd zijn moeten voorkomen. Deze groeperingen moeten in de definitie van een datatype voorkomen of in een *xsd:group*-element, waarmee we zo een groepering globaal kunnen definiëren. Een xsd:group-element is ook een XML Schema particle.

  De waarde van een nOccurs-attribuut moet een XPath 1.0 expressie zijn. Deze expressie zal over de tot dan opgebouwde BSD geëvalueerd worden en moet resulteren in een numerieke waarde die dan het aantal herhalingen bepaalt. Een XPath expressie wordt ten opzichte van een context node geëvalueerd, in dit geval is dat het element dat het XML Schema particle bevat waaraan het nOccurs-attribuut is verbonden.

- **if**

  Het if-attribuut bevat net als het nOccurs-attribuut een XPath expressie die op dezelfde manier geëvalueerd wordt. De expressie moet nu echter evalueren naar een booleanwaarde. Dit bepaalt dan of het XML Schema particle al dan niet uit de bitstream gelezen moet worden, en dus in de BSD opgenomen moet worden.

- **ifNext**

  Het ifNext-attribuut is net als het if-attribuut een conditionele voorwaarde voor een XML Schema particle. Het kan één of twee hexadecimale strings bevatten. Als het één waarde bevat wordt het particle alleen maar gelezen als de volgende bytes in de bitstream gelijk zijn aan de hexadecimale string. Als het twee strings bevat dan stellen deze een bereik voor en wordt het volgende particle alleen maar ingelezen als de volgende bytes in dit bereik liggen. Het aantal bytes dat gelezen moet worden, wordt bepaald door de lengte van de string.

Enkele attributen uit deze uitbreidingen bepalen het aantal herhalingen van een XML Schema particle. Met de XML Schema standaard attributen *minOccurs* en *maxOccurs* kunnen we in een schema aanduiden hoeveel keer een particle mag voorkomen. Deze attributen worden niet gebruikt door een BSDL-parser. Als we de BSD willen valideren met behulp van een standaard

```
<xsd:complexType name="LengthPrependedString">
  <xsd:sequence>
    <xsd:element name="length" type="xsd:unsignedInteger" />
    <xsd:element name="string">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:annotation>
            <xsd:appinfo>
              <bs2:length value="../length" />
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Listing 3.3: Een datatype definitie voor een string waarvan de lengte in een unsigned integer veld voor de string is gegeven.

XML Schema parser ten opzichte van het BS Schema, dan moet het aantal herhalingen van een particle dat door de attributen uit de structurele uitbreidingen is bepaald, wel liggen tussen de grenzen die door het minOccurs- en het maxOccurs-attribuut zijn vastgelegd. Anders zal de standaard XML Schema parser de BSD niet valideren ten opzichte van het BS Schema.

De tweede categorie van uitbreidingen zijn van toepassing op datatypes. Niet alle datatypes kunnen vanzelfsprekend van een bitstream ingelezen worden. Verschillende datatypes hebben een onbepaalde lengte en deze moet bepaald kunnen worden tijdens het BSD-generatie-proces. Tijdens het bitstream-generatie-proces is dit uiteraard niet nodig want dan hebben we de waarde van het element in onze BSD staan. De BSDL-2 uitbreidingen voorzien een aantal facetten om de lengte van zulke datatypes te bepalen. Aangezien de XML Schema standaard niet toelaat eigen facetten toe te voegen, gebruikt de BSDL het annotatie-mechanisme om de uitbreidingen toe te voegen. De nieuwe uitbreidingen hiervoor zijn:

- **length**
  Met het length-element kan de lengte van een datatype bepaald worden door middel van een XPath expressie. Net zoals bij het nOccurs-attribuut wordt deze expressie over de tot dan opgebouwde BSD geëvalueerd en moet ze resulteren in een numerieke waarde. In Listing 3.3 wordt een datatype voor een string gegeven. Eerst wordt een unsigned integer veld met de lengte van de string ingelezen, daarna wordt deze waarde gebruikt om de lengte van de string te bepalen.

- **startCode**
  Het startCode-element kan één of twee hexadecimale waarden bevatten. Als het één

```
<xsd:simpleType name="NullTerminatedString">
  <xsd:restriction base="xsd:string">
    <xsd:annotation>
      <xsd:appinfo>
        <bs2:startCode value="00" />
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:restriction>
</xsd:simpleType>
```

Listing 3.4: Een datatype definitie voor een null-terminated string.

waarde bevat wordt er in de bitstream naar deze waarde gezocht. Als het twee waarden bevat dan is dit een bereik en wordt er in de bitstream naar een waarde in dit bereik gezocht. De lengte van het datatype is dan van de huidige positie tot net voor de gevonden waarde. Als er geen waarde wordt gevonden dan is de lengte van het datatype van de huidige positie tot aan het einde van de bitstream. In Listing 3.4 wordt een voorbeeld gegeven van een datatype dat het startCode-element gebruikt om een null-terminated string voor te stellen. De string wordt ingelezen totdat we een byte met als waarde 0 tegenkomen.

- **endCode**
  Het endCode-element werkt net hetzelfde als het startCode-element, alleen nemen we nu de lengte tot net na de gevonden waarde.

In XML Schema kunnen we een nieuw datatype maken door een union van andere simpele datatypes te nemen. Dit betekent dat de waarde van het datatype aan een van de andere datatypes voldoet. Als we een bitstream van een BSD willen genereren dan moet het type van het element door het *xsi:type* attribuut meegegeven worden. Als we een BSD van een bitstream willen genereren hebben we een mechanisme nodig om te kiezen welk type we gaan gebruiken. Dit gebeurt door de *ifUnion* uitbreiding:

- **ifUnion**
  Als we een union-datatype definiëren, specificeren we een lijst van datatypes. Via het annotatie-mechnisme hangen we hier een reeks ifUnion-elementen aan vast. Elk ifUnion-element bevat een XPath expressie die net als bij het if-element moet evalueren naar een boolean waarde. Als de eerste ifUnion expressie evalueert naar true, zal het BSD-generatie-proces het eerste datatype gebruiken, als de tweede evalueert naar true wordt het tweede datatype gebruikt, .... Datatypes waar geen ifUnion-element voor is opgegeven zullen automatisch gekozen worden als alle voorgaande datatypes in de union niet gekozen zijn. Listing 3.5 demonstreert hoe tussen drie binaire representaties voor één veld gekozen kan worden aan de hand van een op voorhand ingelezen waarde.

```
<xsd:complexType = "UnionDefinition">
  <xsd:sequence>
    <xsd:element name="type" type="xsd:byte" />
    <xsd:element name="UnionDatatype">
      <xsd:simpleType>
        <xsd:union memberTypes="xsd:short xsd:integer xsd:long">
          <xsd:annotation>
            <xsd:appinfo>
              <bs2:ifUnion value="../type == 1" /><!-- use short -->
              <bs2:ifUnion value="../type == 2" /><!-- use integer -->
              <!-- else use long -->
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:union>
      </xsd:simpleType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Listing 3.5: Een datatype definitie dat een union van andere datatypes is.

### BSD-generatie uit een bitstream en een BS Schema

Eens we een BS Schema hebben waar de BSDL-2 uitbreidingen, en eventueel de BSDL-1 uitbreidingen, in zijn verwerkt waar nodig is, kunnen we dit gebruiken om een BSD te genereren van een bitstream. Het BS Schema definieert de bitstream als een hiërarchische structuur. Het proces om een BSD te genereren bestaat eruit om de binaire data om te zetten naar deze structuur.

We gaan nu ter illustratie beschrijven hoe de BSD in Listing 3.1 uit de bitstream in Figuur 3.2 met behulp van een BS Schema is geconstrueerd. De bitstream is een JPEG2000-afbeelding en om hier een BSD van te genereren hebben we een BS Schema nodig dat de syntax van dit bestandsformaat beschrijft. Dit BS Schema beschrijft hoe de boomstructuur in de BSD is gestructureerd en is eveneens ook de koppeling tussen de boomstructuur en de binaire data in de bitstream. In Listing 3.6 en Listing 3.7 zijn de elementdeclaraties uit het gebruikte schema getoond die de elementen in Listing 3.1 definiëren.

We beginnen de constructie van de BSD door het wortelelement te zoeken in het schema. Dit wordt in het volledige schema gegeven door het BSDL-2 *rootElement*-attribuut. Eens we het wortelelement hebben gevonden, heben we de definitie van onze boomstructuur en gaan we deze op een diepte-eerst methode overlopen. In het gebruikte schema is het wortelelement het `Codestream`-element. Het *rootElement*-attribuut waar deze verwijzing in staat wordt hier niet getoond. We voegen dit element toe aan de BSD die we genereren en we verwerken de kinderen ervan. Zoals we in Listing 3.6 kunnen zien bevat dit element een sequentie van

drie elementdeclaraties: het `MainHeader`-, het `Tile`- en het `EOC`-element. De eerste twee elementdeclaraties zijn lokale elementdeclaraties die verwijzen naar een globaal gedefinieerd datatype. De derde elementdeclaratie verwijst naar een globale elementdeclaratie. Geen enkel van deze elementen heeft attributen uit de structurele uitbreidingen in de BDSL-2 namespace. Dit betekent dat elk element exact één keer voorkomt. De declaraties komen voor in een *xsd:sequence*, dus we verwerken ze in de volgorde zoals ze hierin gedeclareerd zijn. De definitie van het `MainHeader`-element wordt hier niet getoond wegens plaatsgebrek, het is voldoende om te weten dat hier enkele velden in staan die informatie geven over hoe de bitstream is geëncodeerd. Als we de declaratie van het `Tile`-element verwerken komen we weer een sequentie tegen, maar deze keer is één van de kinderen een *xsd:choice*-constructie. De XML Schema standaard zegt dat er exact één element uit de choice-constructie in de ouderlijke constructie voorkomt [Con04]. We gaan daarom één voor één controleren als we de elementen kunnen inlezen, zo niet gaan we verder met het volgende. Als we een element hebben kunnen inlezen uit de bitstream schrijven we dit naar de BSD en stoppen we. Zoals eerder besproken in deze sectie, kunnen we elementdeclaraties voorwaardelijk maken met het *if*- of het *ifNext*-attribuut; of we kunnen ook het *nOccurs*-attribuut gebruiken met een XPath expressie en kijken als deze naar een nulwaarde evalueert of niet. Binnen het `Tile`-element hebben we de keuze tussen een `Bitstream`- of een `BitstreamNoSOP`-element. Het `Bitstream`-element is voorwaardelijk gemaakt door middel van een XPath expressie in het *if*-attribuut. We voeren deze uit over de tot nu toe gegenereerde BSD om te kijken als we het element moeten verwerken of niet, als de expressie naar onwaar evalueert gaan we verder met het volgende element. Het volgende element is het `BitstreamNoSOP`-element, dit is niet voorwaardelijk en kunnen we dus direct verwerken. Voor de bitstream die we hier aan het analyseren zijn, evalueert de voorgaande XPath expressie naar waar en gebruiken we dus het `Bitstream`-element. De XPath expressie controleert als de waarde van een veld uit het `MainHeader`-element gelijk is aan 1. Dit veld duidt aan hoe de geëncodeerde data in de bitstream is opgedeeld.

Als we eens alle sequentie- en keuze-constructies hebben verwerkt, komen we uit bij de declaraties met een simpel datatype. De XML Schema standaard definieert simpele datatypes als datatypes zonder kinderen [Con04]. Het zijn de declaraties met deze datatypes die de elementen genereren die de bladeren van onze boomstructuur zullen vormen. Deze elementen bevatten of verwijzen naar de binaire data uit de bitstream. De hoger gelegen elementen die we tot nu toe hebben verwerkt zijn slechts groeperingen van andere elementen die de structuur van het bestandsformaat reflecteren. De eerste declaratie die we tegenkomen met een simpel datatype is het `Marker`-element. Dit is een hexadecimale string die twee bytes in beslag neemt, we lezen nu dus twee bytes in en encoderen deze als een hexadecimale string om als waarde aan het `Marker`-element in de BSD te geven. Voor de volgende twee elementen

`LMarker` en `Nsop` lezen we ook elk twee bytes in maar nu encoderen we ze als een unsigned short waarden. Vervolgens komen we het `PacketData`-element tegen, dit heeft het *byteRange*-datatype uit de BSDL-1 uitbreidingen. De lengte van dit datatype is onbepaald en om deze te weten te komen moeten er BSDL-2 uitbreidingen aan het datatype gekoppeld zijn. In dit geval zijn dat twee *startCode*-elementen. We gaan nu in de bitstream zoeken naar deze waarden en nemen als lengte voor het datatype de huidige positie in de bitstream tot aan het begin van de gevonden code. Het `Packet`-element gebruikt een structurele uitbreiding uit de BSDL-2 namespace. Aan het *ifNext*-attribuut is een hexadecimale string met vier karakters meegegeven. Twee hexadecimale karakters stellen één byte voor en dus gaan we het `Packet`-element inlezen zolang de waarde van de volgende twee bytes gelijk is aan de waarde van het attribuut. De waarde die het einde van een pakket kan aanduiden is de startwaarde van een volgend pakket (FF91) of de waarde die het einde van de bitstream aanduidt (FFD9).

#### Bitstream-generatie uit een BSD en een BS Schema

Het proces om een bitstream te genereren uit een BSD en een BS Schema is veel simpeler. We moeten hierbij namelijk niet het aantal herhalingen van elk element bepalen of de lengtes van sommige datatypes. Deze informatie zit namelijk allemaal al in de BSD zelf.

Stel dat we nu een bitstream willen genereren uit de BSD in Listing 3.1. We moeten de boomstructuur van deze BSD gewoon in diepte-eerst methode overlopen. Voor elk bladelement dat we dan tegenkomen zoeken we het overeenkomstige datatype in het BS Schema zodat we weten hoe we de tekstuele waarde terug in een binaire waarde moeten omzetten. Voor bijvoorbeeld het `Marker`-element zou dit resulteren in twee bytes met als waarde 255 (FF) en 145 (91). Voor het `PacketData`-element, dat van het BSDL-1 datatype *byteRange* is, zou dit betekenen dat we in de originele bitstream vanaf positie 155 435 bytes moeten inlezen en deze wegschrijven naar de nieuwe bitstream.

### 3.1.2 generic Bitstream Syntax Description

BS Schema's bieden een generieke manier om de syntax van een bitstream te beschrijven. Twee problemen die we tegenkomen bij het gebruik van deze schema's echter zijn dat we het schema nodig hebben om opnieuw een bitstream van de BSD te genereren en dat we voor elk formaat een ander BS Schema nodig hebben. Het *generic Bitstream Syntax Schema (gBS Schema)* biedt hier een oplossing voor door een specifiek schema te standaardiseren [ISO03]. Het schema maakt gebruikt van de BSDL-1 uitbreidingen en is zo ontwikkeld dat de meeste bitstreams er op een zo optimaal mogelijke manier mee beschreven kunnen worden. Er zijn slechts twee generieke datatypes in beschreven die toelaten om bitstreams op een hiërarchische wijze voor te stellen. Het aantal mogelijke element-types is vast gedefinieerd dus biedt het

```
<xsd:element name="Codestream" type="jp2:CodestreamType"/>

<xsd:complexType name="CodestreamType">
  <xsd:sequence>
    <xsd:element name="MainHeader" type="jp2:MainHeaderType"/>
    <xsd:element name="Tile" type="jp2:TileType"/>
    <xsd:element ref="jp2:EOC"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="EOC" bs2:ifNext="FFD9">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Marker" type="jp2:MarkerType" fixed="FFD9"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="TileType">
  <xsd:sequence>
    <xsd:element name="TileHeader" type="jp2:TileHeaderType"/>
    <xsd:choice>
      <xsd:element name="Bitstream" type="jp2:BitstreamType"
          bs2:if="/jp2:Codestream/jp2:MainHeader/jp2:COD/
                  jp2:Scod/jp2:SOPMarkersUsed = 1"/>
      <xsd:element name="BitstreamNoSOP" type="jp2:PacketDataType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BitstreamType">
  <xsd:sequence>
    <xsd:element name="Packet" maxOccurs="unbounded" bs2:ifNext="FF91">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="jp2:SOP" minOccurs="0"/>
          <xsd:element name="PacketData" type="jp2:PacketDataType"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="SOP" bs2:ifNext="FF91">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Marker" type="jp2:MarkerType"/>
      <xsd:element name="LMarker" type="jp2:LMarkerType"/>
      <xsd:element name="Nsop" type="xsd:unsignedShort"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Listing 3.6: Datatypes uit het JPEG2000 BS Schema.

```
<xsd:simpleType name="MarkerType" id="MarkerTypeID">
  <xsd:restriction base="xsd:hexBinary">
    <xsd:length value="2"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="LMarkerType">
  <xsd:restriction base="xsd:unsignedShort"/>
</xsd:simpleType>

<xsd:simpleType name="PacketDataType">
  <xsd:restriction base="bs1:byteRange">
    <xsd:annotation>
      <xsd:appinfo>
        <bs2:startCode value="FF91"/>
        <bs2:startCode value="FFD9"/>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:restriction>
</xsd:simpleType>
```

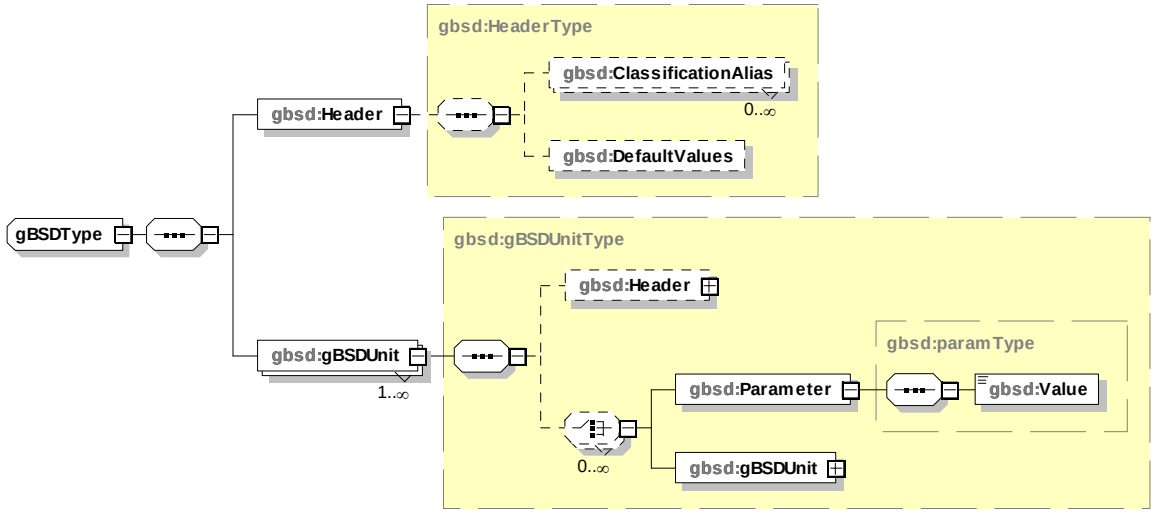
Listing 3.7: Vervolg datatypes uit het JPEG2000 BS Schema.

gBS Schema de mogelijkheid om semantische informatie aan elementen te hangen via een speciaal attribuut. Applicaties of het transformatieproces kunnen dan van deze semantische informatie gebruik maken. Een BSD die voldoet aan het gBS Schema wordt een *generic Bitstream Syntax Description (gBSD)* genoemd. Aangezien we nu weten welke elementen, en wat hun binaire representatie is, er in een gBSD mogen voorkomen, hebben we het schema niet meer nodig om er een bitstream van te genereren. De processor die gBSDs omzet naar bitstreams kan dan ook geöptimaliseerd worden en is nu helemaal formaatonafhankelijk. De abstractheid en het generieke karakter van het gBS Schema heeft ook enkele nadelen. Het schema kan niet gebruikt worden om automatisch een gBSD van een bitstream te genereren omdat de datatypes ons niet van de nodige formaatspecifieke informatie voorzien. gBSDs worden meestal gecreëerd door met een BS Schema voor het formaat van de bitstream een BSD te genereren en deze dan te transformeren naar een gBSD. Doordat het gBS Schema zo algemeen is zijn gBSD-beschrijvingen van een bitstream ook typisch groter van omvang als BSDs die voldoen aan een schema dat specifiek is afgestemd op het formaat van de bitstream.

### De gBSD datatypes

In Figuur 3.4 is de hiërarchie van de datatypes in het gBS Schema afgebeeld. De betekenis van de datatypes en hun attributen is als volgt [ISO03]:

- **addressAttributes**
  Er zijn twee datatypes die naar binaire data in de bitstream verwijzen in het gBS




Figuur 3.4: De gBSD datatypes hiërarchie

Schema. Ze zijn beiden afgeleid van het BSDL-1 bitstreamSegment-datatype waardoor er een start en een lengte waarde kan opgegeven worden voor het binair segment waar het naar verwijst. addressAtributes is een groep van attributen waarmee gespecificeerd kan worden hoe de waarden hiervan geïnterpreteerd moeten worden. De groep bestaat uit drie attributen:

- *addressMode*
  Het addressMode-attribuut bepaalt hoe de startwaarde geïnterpreteerd moet worden. Het kan drie waarden hebben: *Absolute* betekent dat de startwaarde ten opzichte van het begin van de bitstream genomen moet worden; *Consecutive* betekent dat de startwaarde begint waar het voorgaande element eindigde, de startwaarde moet dus niet opgegeven worden; *Offset* betekent dat de startwaarde ten opzichte van het einde van het vorig element genomen moet worden.
- *addressUnit*
  Het addressUnit-attribuut bepaalt de eenheid van de startwaarde en de lengtewaarde. Het kan de waarden *bit* of *byte* bevatten.
- *globalAddressInfo*
  Het globalAddressInfo-attribuut is een URI naar de bitstream waar de binaire data tot behoort.

- **Header**
  Het Header-element kan enkele ClassificationAlias-elementen bevatten die elk een alias geven aan de URI van een classificatieschema. De syntax van classificatieschema's is

gedefinieerd in de ISO/IEC 15938-5 (MPEG-7) standaard [ISO02]. Een classificatieschema is een gecontrolleerd vocabularium met een *identifier* voor elke term. Elementen of attributen in de gBSD kunnen dan naar deze termen refereren door het alias van het vocabularium en de identifier van de term te gebruiken. Het Header-element kan ook een DefaultValues-element bevatten dat de addressAttributes-groep bevat. Het Header-element wordt gebruikt om de standaardwaarden voor de addresserings-attributen voor alle afstammelingen van zijn ouderlijk element te bepalen.

- **gBSDUnit**
  Het gBSDUnit-element is één van de twee datatypes in het gBS Schema dat naar binaire data in de bitstream verwijst. Het is afgeleid van het BSDL-1 bitstreamSegment-datatype en heeft hierdoor de attributen start en length om aan te duiden waar de data van dit blok zich in de originele bitstream bevindt. Het gBSDUnit-element kan zowel Parameter- als andere gBSDUnit-elementen bevatten waardoor dit het element is waar de bitstream hiërarchisch mee voorgesteld kan worden.

  Het gBSDUnit-element kan attributen uit de groep addressAttributes bevatten die de addresseringsmethode voor het element zelf bepalen. Een gBSDUnit-element kan ook een *syntacticalLabel*-attribuut hebben dat de semantiek van het logische blok, dat door dit element wordt voorgesteld, aangeeft. De waarde van het attribuut verwijst naar een term uit één van de classificatieschema's die in de bovenliggende Header-elementen zijn gedeclareerd. Applicaties die bewust zijn van het gebruikte classificatieschema of het gBSD-transformatie-proces kunnen dan gebruik maken van deze waarde. Met dit attribuut kunnen we bijvoorbeeld aanduiden dat een gBSDUnit een B-frame uit een MPEG4 videostream voorstelt. Een gBSDUnit-element kan een *marker*-attribuut hebben waarmee we door middel van een gewone stringwaarde dit blok kunnen markeren. Dit attribuut kunnen we gebruiken om bijvoorbeeld in een film bepaalde scenes te markeren als geweldadig door de frames van deze scene in een gBSDUnit-element te groeperen en dit element dan te markeren.

- **Parameter**
  Het Parameter-element is ook van het BSDL-1 bitstreamSegment-datatype afgeleid en is daarmee het tweede datatype dat naar binaire data in de bitstream verwijst. Het Parameter-element heeft exact één Value-element met hierin een tekstuele representatie van de data waar het naar verwijst. Doordat we een tekstuele representatie van de data in de gBSD opnemen kunnen we deze wijzigen tijdens een transformatie. Om de tekstuele waarde terug om te kunnen zetten naar binaire data moeten we het type van het Value-element meegegeven via het *xsi:type*-attribuut, zodat we weten hoeveel bytes we nodig hebben en hoe we de tekstuele waarde terug moeten encoderen. Het Parameter-

element kan net zoals het gBSDUnit-element attributen uit de groep addressAttributes en het marker-attribuut bevatten. Het element kan ook een *name*-attribuut hebben dat dezelfde betekenis heeft als het syntacticalLabel-attribuut van een gBSDUnit.

In Listing 3.8 wordt een uittreksel van een gBSD gegeven. De gBSD beschrijft dezelfde JPEG2000-afbeelding als de afbeelding die door de BSD in Listing 3.1 beschreven wordt. De gBSD is gecreëerd door deze BSD met een XSLT-stylesheet te transformeren. In de gBSD wordt de formaatspecifieke informatie gegeven door het syntacticalLabel- en name-attribuut van respectievelijk het gBSDUnit- en het Parameter-element. Aangezien we alleen deze twee datatypes in de gBSD mogen gebruiken en niet specifiek gedefinieerde die op het formaat zijn afgestemd, is de gBSD ook veel groter in omvang dan de BSD. Groeperende elementen en elementen die naar binaire data verwijzen uit de BSD zijn vervangen door gBSDUnit-elementen. Elementen die een tekstuele representatie van de binaire data bevatten zijn vervangen door Parameter-elementen. De stylesheet heeft voor elk JPEG2000-pakket ook een waarde voor het marker-attribuut gegenereerd. Aan de hand van deze waarde kan een transformatiestylesheet weten welke pakketten er weggelaten moeten worden om de spatiale resolutie aan te passen of het aantal kwaliteitslagen te minderen. Dit wil zeggen dat de transformatie bewust moet zijn van de betekenis van de markers. Als we een adaptatie-engine helemaal formaat-onafhankelijk willen maken dan moeten we de transformatiestylesheet in het Digital Item aan de gBSD koppelen.

## 3.2 Adaptation Quality of Service

De BSD Tools adapteren multimedia bronnen door het weglaten van datasegmenten en simpele wijzigingen in de bitstream door te voeren. Door het simpele karakter van deze wijzigingen zijn er maar een beperkt aantal adaptaties beschikbaar. De adaptatie van een multimedia bron verloopt door een XML transformatie uit te voeren op de BSD van de bitstream en deze te reflecteren op de bitstream zelf. De *Adaptation Quality of Service Tools (AdaptationQoS Tools)* sturen het adaptatieproces door restricties op een aantal adaptatieoperatoren en de resulterende kwaliteit hiervan te mappen [ISO03]. Restricties zijn de mogelijkheden van de *terminal* en het netwerk waarvoor de bron wordt aangepast. Dit kunnen eigenschappen zijn zoals de bandbreedte van het netwerk, de spatiale resolutie van het ontvangstapparaat of de maximale processortijd dat het ontvangstapparaat kan besteden aan het afspelen van de bron. Een stel van adaptatieoperatoren samen vormen een adaptatieoperatie. Adaptatieoperatoren kunnen de parameters zijn die gebruikt worden tijdens het XML transformatie proces. Een adaptatieoperator kan bijvoorbeeld zijn hoeveel spatial layers er gebruikt moeten worden of hoeveel bitplanes er moeten gebruikt worden. Een AdaptationQoS-beschrijving kan met een

```
<DIA>
  <Description xsi:type="gBSDType">
    <Header>
      <ClassificationSchemeAlias alias="J2K"
        href="urn:jpeg:jpeg2000:cs:syntacticalLabels"/>
    </Header>
    <gBSDUnit syntacticalLabel=":J2K:MainHeader"
      start="0" length="135">
    <!-- JPEG2000 MainHeader elements -->
    </gBSDUnit>
    <gBSDUnit syntacticalLabel=":J2K:Tile"
      start="135" length="240232">
      <gBSDUnit syntacticalLabel=":J2K:TileHeader"
        length="14" addressMode="Consecutive">
      <!-- JPEG2000 TileHeader elements -->
      </gBSDUnit>

      <!-- Stylesheet maakt geen gBSDUnit voor het
        JPEG2000 Bitstream element -->
      <gBSDUnit syntacticalLabel=":J2K:Packet"
        start="149" length="441" marker="L0 R0 C0 P0">
        <gBSDUnit addressMode="Consecutive" length="441">
          <!-- JPEG2000 Marker element -->
          <gBSDUnit length="2"/>
          <!-- JPEG2000 LMarker element -->
          <Parameter name=":J2K:LMarker" length="2">
            <Value xsi:type="xs:unsignedShort">4</Value>
          </Parameter>
          <!-- JPEG2000 Nsop element -->
          <Parameter name=":J2K:Nsop" length="2">
            <Value xsi:type="xs:unsignedShort">0</Value>
          </Parameter>
          <!-- JPEG2000 PacketData element -->
          <gBSDUnit length="435"/>
        </gBSDUnit>
      </gBSDUnit>

      <!-- ... more packets ... -->
    </gBSDUnit>
    <gBSDUnit syntacticalLabel=":J2K:EOC" start="240367" length="2"/>
  </Description>
</DIA>
```

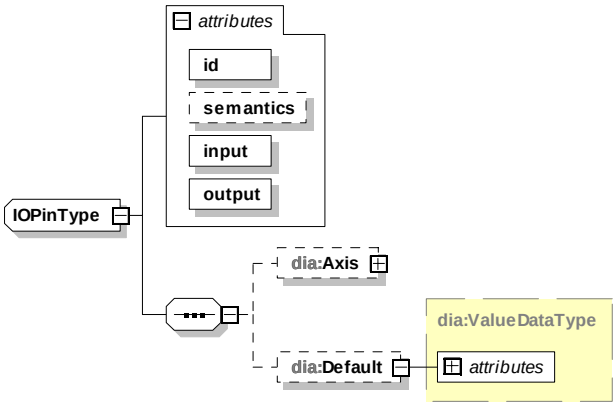
Listing 3.8: Deel uit de gBSD van een JPEG2000-bestand.

adaptatieoperatie ook een kwaliteitswaarde (*Utility*) associeren. De AdaptationQoS Tools helpen ons dus met de optimale QoS te berekenen voor enkele gegeven restricties.

In de AdaptationQoS Tools wordt een flexibel en uitbreidbaar ontwerp voor datatypes gebruikt [ISO03]. Er zijn drie soorten datatypes gedefinieerd: een *ValueDataType* of een enkelvoudige waarde, een *VectorDataType* is een lijst van enkelvoudige waarden en een *MatrixDataType* is een multidimensionale tabel van enkelvoudige waarden. Het zijn alle drie abstracte basisklasses waar gemakkelijk specifieke implementaties van afgeleid kunnen worden. MPEG-21 specificeert zelf voor elk soort drie implementaties, elk van het type integer, float en string.

Een AdaptationQoS-beschrijving is een opeenvolging van een optioneel *Header*-element gevolgd door een reeks van *Module*-elementen en *IOPin*-elementen. Het Header-element kan een aantal ClassificationAlias-elementen bevatten die elk een alias geven aan de URI van een classificatieschema. De Module-elementen zijn de bouwstenen van de AdaptationQoS Tool en de IOPin-elementen vormen de input- en output-interface.

### 3.2.1 IOPins




Figuur 3.5: Het IOPin-datatype

IOPins kan je vergelijken met globale variabelen. Ze binden de modules aan mekaar en zijn tegelijkertijd ook de input- en output-interface voor de AdaptationQoS Tool. De waarde van een IOPin kan dus door een externe applicatie ingevuld en/of gelezen worden of ze kan als een tijdelijke variabele door een Module ingevuld en/of gelezen worden.

Een IOPin wordt door de volgende attributen en kinderelementen gedefinieerd:

- **id**
  Het id-attribuut bevat een waarde die voor het ganse document uniek is. De waar-

de identificeert de IOPin op een unieke wijze en laat modules toe naar een IOPin te verwijzen.

- **semantics**
  Het semantics-attribuut kan optioneel naar een term uit een classificatieschema verwijzen om een semantische betekenis aan de IOPin te geven. Applicaties die van het classificatieschema bewust zijn kunnen de waarde van de IOPin dan naar behoren invullen of uitlezen en interpreteren.

- **input** en **output**
  Met het input- en output-attribuut kan aangeduid worden als een IOPin als respectievelijk input- en/of output-waarde gebruikt wordt.

- **Axis**
  Een IOPin kan een globale as van een LookUpTable-module bevatten. Meer informatie hierover volgt in Sectie 3.2.2.

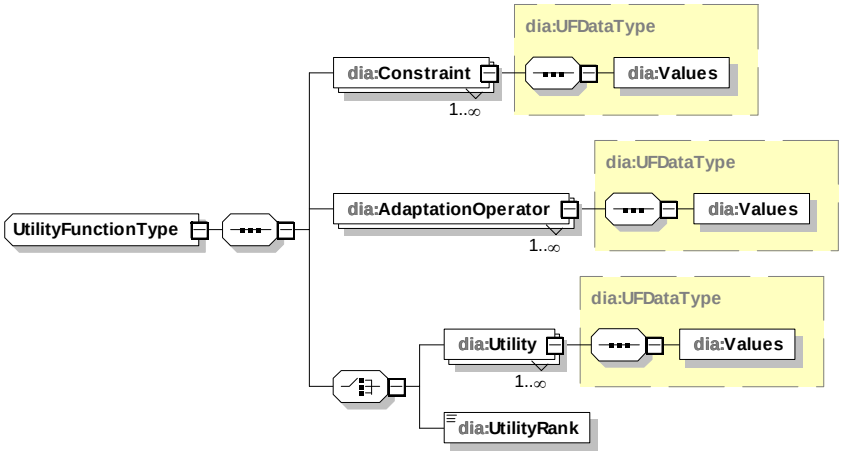
- **Default**
  Het Default-element kan een standaardwaarde voor een IOPin bevatten. De standaardwaarde wordt voor de IOPin gebruikt als de modules anders geen waarde voor de IOPin kunnen bepalen.

### 3.2.2 Modules

Modules zijn de bouwstenen van een AdaptationQoS beschrijving. Ze gedragen zich als functies in de zin dat ze aan de hand van waarden van IOPins de waarden van andere IOPins berekenen. Er zijn drie soorten modules gedefinieerd [ISO03, BPdWK06], het type dat gebruikt wordt moet aangeduid worden door het *xsi:type* attribuut bij het Module-element te bruiken.

#### Utility function

In Figuur 3.6 is de structuur van de UtilityFunction-module afgebeeld. De UtilityFunction-module bevat een reeks van *Constraint*-, *AdaptationOperator*- en *Utility*-elementen. Alle drie bevatten ze een lijst van waarden en zijn ze gekoppeld aan een IOPin, die ook de semantiek van het element bepaalt. Een UtilityFunction-module associeert een Constraint-punt met een aantal adaptatieoperatoren en de kwaliteit die resulteert van deze adaptatieoperatoren toe te passen (Utility). In een Utility-vector stoppen we waarden die de resulterende kwaliteit van een adaptatieoperatie representeren. Dit kan bijvoorbeeld het *Signal to Noise Ratio (SNR)* van een afbeelding zijn. In de plaats van een Utility-vector kunnen we een UtilityRank gebruiken. Dit is een lijst met strict postieve integer waarden. Deze waarden geven geen




Figuur 3.6: De UtilityFunction-module

numerieke waarde aan de resulterende kwaliteit, maar dienen alleen maar om een rangorde aan de verschillende adaptatieoperaties te geven.

In Listing 3.9 wordt het gebruik van een UtilityFunction-module gedemonstreerd. We gaan een JPEG2000-afbeelding adapteren voor een systeem dat een restrictie legt op de bestandsgrootte van de afbeelding. Er zijn twee adaptatieoperatoren waarmee we de bestandsgrootte van de afbeelding kunnen verkleinen: we kunnen de *spatiale resolutie* een aantal keer halveren en we kunnen het aantal *kwaliteitslagen* reduceren. De afbeelding heeft een resolutie van $768 \times 512$ en is geëncodeerd met vijf wavelet-decomposities, dit betekent dat we de resolutie tot vijf keer toe kunnen halveren, de laagste resolutie die we kunnen verkrijgen is dus $24 \times 16$. De afbeelding is ook geëncodeerd met vijf kwaliteitslagen die we kunnen weg laten, per laag dat we weg laten wordt het signal-to-noise ratio lager. De gegeven AdaptationQoS-beschrijving mapt de toegelaten bestandsgrootte op een adaptatieoperatie die uit deze twee adaptatieoperatoren bestaat. De input voor de beschrijving wordt gegeven door de IOPin `FILESIZE`, dit is de toegelaten bestandsgrootte die de afbeelding mag hebben. Deze waarde wordt door de UtilityFunction-module gemapt op een lijst van acht bestandsgroottes die we door adaptaties kunnen verkrijgen. Elke bestandsgrootte is gekoppeld aan de twee operatoren die we moeten toepassen om deze te bereiken. Deze operatoren zijn de output van de AdaptationQoS-tool, ze zijn gekoppeld aan de IOPins `SCALE` en `SNRLAYERS` die respectievelijk staan voor de reductie in spatiale resolutie en het aantal gebruikte kwaliteitslagen. De bestandsgrootte van de originele afbeelding is 235kB. Voor deze grootte te bereiken gaan we de resolutie nul keer halveren en gebruiken we alle vijf de kwaliteitslagen, er gebeurt dus geen aanpassing. Als de bestandsgrootte van de afbeelding kleiner moet zijn dan gaan we dit eerst proberen te doen door kwaliteitslagen weg te laten, als dit niet lukt gaan we de resolutie halveren en terug pro-

```
<DIA>
  <Description xsi:type="AdaptationQoSType">
    <Module xsi:type="UtilityFunctionType">
      <Constraint iOPinRef="FILESIZE">
        <Values xsi:type="IntegerVectorType">
          235 187 148 110 70 43 27 13
        </Values>
      </Constraint>
      <AdaptationOperator iOPinRef="SCALE">
        <Values xsi:type="IntegerVectorType">
          0 0 0 1 1 2 2 3
        </Values>
      </AdaptationOperator>
      <AdaptationOperator iOPinRef="SNRLAYERS">
        <Values xsi:type="IntegerVectorType">
          5 4 3 5 3 5 2 4
        </Values>
      </AdaptationOperator>
      <UtilityRank>1 2 3 4 5 6 7 8</UtilityRank>
    </Module>
    <IOPin id="FILESIZE" input="true" output="false" />
    <IOPin id="SCALE" input="false" output="true" />
    <IOPin id="SNRLAYERS" input="false" output="true" />
  </Description>
</DIA
```

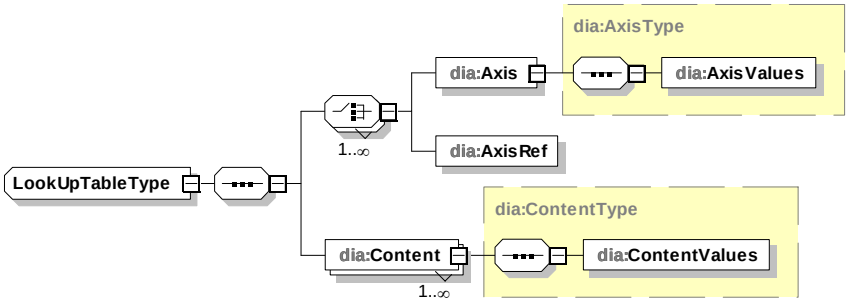
Listing 3.9: Voorbeeld van het gebruik van een UtilityFunction-module.

beren zoveel mogelijk kwaliteitslagen te gebruiken. We gebruiken geen numerieke waarde die de kwaliteit van de adaptatie reflecteert. In de plaats hiervan gebruiken we een UtilityRank die een rangorde geeft aan de mogelijke adaptatieoperaties.

### Look-up table

De LookUpTable-module wordt gebruikt voor een dichte, discrete data representatie. Er wordt een mutidimensionale tabel gebruikt om de data voor te stellen. De structuur van het moduletype is afgebeeld in Figuur 3.7. De module wordt beschreven door de assen van de tabel en de inhoudswaarden van de tabel. Zowel de assen als de inhoudswaarden zijn gekoppeld aan een IOPin. De waarden van de IOPins die aan de assen gekoppeld zijn, worden op de assen gemapt om zo de juiste inhoudswaarde van de tabel te bepalen en deze naar de IOPin weg te schrijven die hieraan gekoppeld is.

De assen van de tabel worden gegeven door *Axis*- of *AxisRef*-elementen. Axis-elementen bevatten een definitie van een as en een koppeling met een IOPin. AxisRef-elementen bevatten een verwijzing naar een IOPin met een globale asdefinitie in. Een asdefinitie bevat een lijst van waarden die de punten op de as voorstellen. Met het *roundingMethod*-attribuut van de




Figuur 3.7: De LookUpTable-module

asdefinitie kunnen we bepalen hoe we de juiste inhoudswaarde van de tabel berekenen als de IOPin waarde niet samenvalt met een punt op de as. Het attribuut bevat een verwijzing naar een term uit een classificatieschema. MPEG-21 DIA standaardiseert een schema met vier afrondingsmethodes in: *floor* is de standaardafrondingsmethode, de index van het punt met de grootste waarde kleiner dan die van de IOPin wordt in de tabel gebruikt; *ceil*, de index van het punt met de kleinste waarde groter dan die van de IOPin wordt in de tabel gebruikt; *linear*, er wordt lineair geinterpoleerd tussen de inhoudswaarden die overeenkomen met de punten waar de IOPin tussen valt en *nearest*, de index van het punt waarvan de waarde het dichtst bij de IOPin ligt wordt in de tabel gebruikt. Aan de asdefinitie kan ook een standaardindex meegegeven worden door het *defaultIndex*-attribuut op te geven.

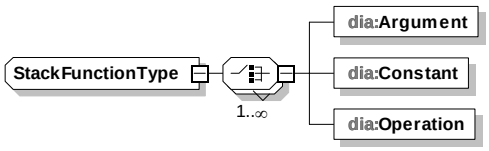
Het Content-element bevat de inhoudswaarden van de tabel. Het element kan een *defaultIndex*-attribuut hebben net als een as dat de standaardindex geeft voor als er een input IOPin waarde niet aanwezig is of de inhoudswaarde niet kan bepaald worden door de mappings van de IOPins op de assen. Het element bevat ook een referentie naar een IOPin die de waarde aanneemt die door de mappings op de assen wordt berekend.

In Listing 3.10 wordt een voorbeeld van een LookUpTable-module gegeven. De module kan een parameter bepalen, voor de afbeelding uit het vorige voorbeeld, die de spatiale resolutie reduceert zodat de afbeelding net op het scherm van de gebruiker past. De input-interface van de module bestaat uit twee IOPins, `SCREENWIDTH` en `SCREENHEIGHT`, waar respectievelijk de schermbreedte en schermhoogte in pixels aan meegegeven wordt. Elke input-waarde wordt in de LookUpTable-module gemapt op een as waar de breedtes of hoogtes op staan waar we de afbeelding naar kunnen adapteren. Voor elke as wordt de standaard afrondingsmethode gebruikt, het punt met de grootste waarde kleiner dan de input-waarde wordt gekozen. Als we de punten op de assen hebben bepaald kunnen we de waarde uit de inhoudstabel halen. Deze waarde geeft ons de parameter om de afbeelding te schalen zodat ze binnen de opgegeven hoogte en breedte valt.

```
<DIA>
  <Description xsi:type="AdaptationQoSType">
    <Module xsi:type="LookUpTableType">
      <Axis iOPinRef="SCREENWIDTH">
        <AxisValues xsi:type="IntegerVectorType">
          24 48 96 192 384 768
        </AxisValues>
      </Axis>
      <Axis iOPinRef="SCREENHEIGHT">
        <AxisValues xsi:type="IntegerVectorType">
          16 32 64 128 256 512
        </AxisValues>
      </Axis>
      <Content iOPinRef="SCALE">
        <ContentValues mpeg7:dim="6 6" xsi:type="IntegerMatrixType">
          <Matrix>
            5 5 5 5 5 5
            5 4 4 4 4 4
            5 4 3 3 3 3
            5 4 3 2 2 2
            5 4 3 2 1 1
            5 4 3 2 1 0
          </Matrix>
        </ContentValues>
      </Content>
    </Module>
    <IOPin id="SCREENWIDTH" input="true" output="false" />
    <IOPin id="SCREENHEIGHT" input="true" output="false" />
    <IOPin id="SCALE" input="false" output="true" />
  </Description>
</DIA>
```

Listing 3.10: Voorbeeld van het gebruik van een LookUpTable-module.

### Stack function




Figuur 3.8: De StackFunction-module

De StackFunction-module wordt gebruikt voor een continue data representatie. De module bevat een lijst van operatoren en operanden die een numerieke functie beschrijven. De lijst is opgesteld in de suffixnotatie, dit betekent dat de operanden boven de operator staan die ze gebruikt. De functie wordt uitgevoerd door van boven naar onder alle operatoren uit te voeren, en de operator en zijn operanden te vervangen door het resultaat.

Een stackfunctie bestaat uit een reeks van *Argument*-, *Constant*- of *Operation*-elementen. Argument-elementen zijn variabele operanden. Ze verwijzen naar een IOPin die de waarde van de operand bepaalt. Een Constant-element is een constante operand, het element bevat de te gebruiken waarde. Een Operation-element is een operator. Het element heeft een *operator*-attribuut dat verwijst naar een term uit een classificatieschema. MPEG-21 heeft een classificatieschema gestandaardiseerd met hierin een dertigtal operatoren in. Het Module-element binnen de AdaptationQoS-beschrijving dat de StackFunction-module bevat moet naar een IOPin verwijzen die het resultaat van de functie als waarde aanneemt.

Stel nu dat we de twee voorgaande voorbeelden, de UtilityFunction-module uit Listing 3.9 en de LookUpTable-module uit Listing 3.10, willen combineren. Hiermee zouden we een AdaptationQoS-beschrijving kunnen maken die een afbeelding aan zowel de toegelaten bestandsgrootte als de grootte van het scherm van de gebruiker kan aanpassen. Er treedt echter een conflict op, beide modules willen namelijk de `SCALE` IOPin invullen. Als we de twee modules in één beschrijving zouden combineren dan zou de uitvoering van de tweede module de berekende waarde door de eerste module overschrijven en zou de eerste nutteloos zijn. In Listing 3.11 wordt een StackFunction-module geïllustreerd die dit probleem kan oplossen. Allereerst passen we de modules uit de voorgaande voorbeelden aan. We laten de UtilityFunction-module een waarde berekenen voor de nieuwe IOPin `SCALE1` en de LookUpTable-module voor de nieuwe IOPin `SCALE2`. Deze nieuwe IOPins functioneren als tijdelijke variabelen die samen met de StackFunction-module de twee modules aan elkaar koppelen. De StackFunction-module heeft als invoer de twee nieuwe IOPins en als uitvoer het maximum van de twee. De uitvoer van de module is de parameter die gebruikt gaat worden voor de uiteindelijke adaptatie en zorgt voor de grootste reductie in spatiale resolutie.

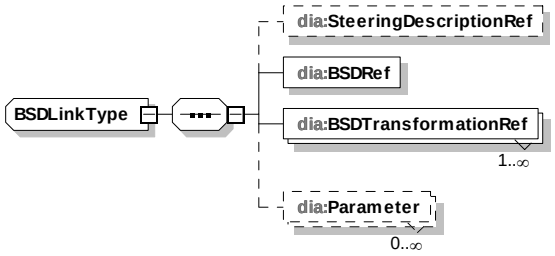
```
<!-- geeft een alias aan het classificatieschema voor operatoren -->
<Header>
  <ClassificationSchemeAlias alias="SF"
    href="urn:mpeg:mpeg21:2003:01-DIA-StackFunctionOperatorCS-NS" />
</Header>

<!-- de StackFunction-module -->
<Module xsi:type="StackFunctionType" iOPinRef="SCALE">
  <Argument iOPinRef="SCALE1" />
  <Argument iOPinRef="SCALE2" />
  <Operation operator=":SF:20" /><!-- max(x, y) operator -->
</Module>
```

Listing 3.11: Voorbeeld van het gebruik van een StackFunction-module.

Hierdoor voldoet de resulterende adaptatie aan beide restricties.

## 3.3 Bitstream Syntax Description Link




Figuur 3.9: De BSDLink-tool

De *Bitstream Syntax Description Link Tool (BSDLink Tool)* beschrijft de link tussen een BSD of gBSD, één of meerdere transformatiestylesheets hiervoor en een *steering description*. Het *SteeringDescriptionRef*-element bevat een referentie naar de steering description [ISO03]. Een steering description is een tool die het adaptatieproces stuurt door te beslissen hoe een bron aangepast moet worden om aan de restricties van het systeem en het netwerk van de gebruiker te voldoen. De steering descriptions die momenteel zijn toegelaten, zijn een AdaptationQoS-beschrijving (zie Sectie 3.2) of het Choice/Selection mechanisme uit de Digitial Item Declaration Language (zie Sectie 2.1.8). Het *BSDRef*-element bevat een referentie naar een een BSD of een gBSD die de binaire structuur van een bron beschrijft. Een *BSDTransformationRef*-element is een referentie naar een stylesheet die de BSD of gBSD kan transformeren. Een stylesheet heeft een aantal parameters die we als invoer kunnen geven om te bepalen hoe de stylesheet de BSD of gBSD zal transformeren. Er kan naar meerdere stylesheets verwezen worden in een BSDLink-beschrijving. De stylesheets moeten

```
<DIA>
  <Description xsi:type="BSDLinkType">
    <SteeringDescriptionRef uri="ScaleLookUpTable.xml" />
    <BSDRef uri="city.xml" />
    <BSDTransformationRef uri="jp2.xsl" />
    <Parameter xsi:type="IOPinRefType" name="scale">
      <Value>SCALE</Value>
    </Parameter>
  </Description>
</DIA>
```

Listing 3.12: Voorbeeld van een BSDLink-beschrijving

dezelfde transformaties op de BSD of gBSD uitvoeren maar mogen in een ander formaat zijn, zodat de applicatie kan beslissen welke stylesheet gebruikt wordt. Er zijn momenteel twee mogelijke stylesheets toegelaten, een XSLT- of een STX-stylesheet. Tenslotte kunnen we met *Parameter*-elementen de link leggen tussen de steering description en de stylesheet(s). Een Parameter-element kan een constante waarde bevatten of kan de output van de steering description mappen op een input parameter van een stylesheet.

In Listing 3.12 wordt een BSDLink-beschrijving getoond die een adaptatie-architectuur vormt voor de BSD uit Listing 3.1: `city.xml`. Deze BSD is een beschrijving van een JPEG-2000 afbeelding en wordt daarom gekoppeld aan een XSLT-stylesheet hiervoor: `jp2.xsl`. Als steering description wordt naar de AdaptationQoS-beschrijving uit Listing 3.10 gerefereerd: `ScaleLookUpTable.xml`. Deze heeft één output IOPin `SCALE` en die wordt gekoppeld aan de input-paramter `scale` van de stylesheet. Deze BSDLink-beschrijving zouden we bijvoorbeeld met een Descriptor-element in een Digital Item declaratie aan de afbeelding kunnen koppelen.

# Hoofdstuk 4

# Implementatie

De implementatie voor deze thesis bestaat uit een Digital Item Declaration Language-parser en een Digital Item Adaptation-engine. De Digital Item Declaration Language-parser is ontwikkeld zodat we onze DIA-tools in een Digital Item kunnen detecteren en onttrekken. In de DIDL-parser zit ook een raamwerk dat het makkelijk maakt om XML-documenten waar het model van de inhoud op voorhand van bekend is, om te zetten naar een hiërarchie van C++ klasses. Het Digital Item Adaptation gedeelte gebruikt dit raamwerk om de beschrijvingen te parsen. De implementatie van de verschillende tools staat los van mekaar. Omdat DIA geen adaptatie-engine standaardiseerd maar alleen de werking van de verschillende tools zijn deze geïmplementeerd in een library. Er is dan een voorbeeld programma geschreven dat de tools in de library gebruikt om een adaptatie-engine mee op te stellen.

## 4.1 Digital Item Declaration Language parser

Om een document in het Digital Item Declaration Language-formaat te parsen is er gebruikt gemaakt van de Xerces-C++ bibliotheek [XER]. Deze bibliotheek maakt het relatief gemakkelijk om in een applicatie XML-documenten te parsen en eventueel te valideren met een bijbehorend XML Schema. De Digital Item Declaration Language parser code is een bibliotheek die gebruikt kan worden om Digital Items naar een interne structuur te vertalen. Eens we deze structuur hebben kunnen we gemakkelijk doorheen Digital Items navigeren en de beschrijvingen zoeken die we nodig hebben, zoals bijvoorbeeld DIA-beschrijvingen.

De Xerces-C++ bibliotheek biedt twee welbekende *Application Programming Interfaces (APIs)* aan om een XML-document mee te parsen. De *Document Object Model (DOM)*-interface is gebaseerd op een boomstructuur die wordt opgebouwd tijdens het parsen [NWS+00]. Deze boomstructuur heeft een heel generiek karakter. Elk element uit het document bevindt zich in de boom en is onder andere verbonden met zijn kinderen, ouder en attributen waardoor

we in de applicatie heel gemakkelijk doorheen de boom kunnen navigeren. Dit maakt het vrij gemakkelijk om documenten te parsen met een DOM-interface. De interface voorziet ook de mogelijkheden om de boomstructuur te wijzigen en deze terug uit te schrijven naar een XML-document. Een nadeel van de DOM-interface is dat het document in één keer ingelezen moet worden om de boomstructuur op te bouwen. Aangezien de interface voor het hele document een boomstructuur in het geheugen opbouwt, groeit het geheugengebruik en de benodigde berekeningstijd snel met de grootte van het XML-document. Een alternatief om een XML-document te parsen is de *Simple API for XML (SAX)* [XMLa]. Deze interface gebruikt geen interne datastructuur maar is een event-gebaseerde API. Dit betekent dat terwijl een document geparset wordt door de API er events naar de applicatie gestuurd worden zoals de start van een element, het einde van een element of tekstuele data die het element bevat. In deze events kunnen we dan onze eigen specifieke datastructuren opbouwen. Hierdoor is het gebruik van SAX sneller en efficiënter in geheugengebruik. Doordat een SAX-parser events doorgeeft als hij ze tegenkomt hoeft het document ook nog niet volledig beschikbaar te zijn als het parsen begint. Het nadeel aan het gebruik van een SAX-parser is dat de code om uitgebreide documenten te parsen al snel complex wordt.

Digital Items zijn ontworpen om multimediabronnen te groeperen en te associeren met metadata. Veel delen van de MPEG-21 standaard standaardiseren ook nog eens tools die in de vorm van extra metadata aan een Digital Item gekoppeld kunnen worden. Hierdoor kan de omvang van een Digital Item al snel grote proporties aannemen. Aangezien de DOM-interface een document in één keer parset en de hele structuur in het geheugen houdt, is er gekozen voor een implementatie die gebruik maakt van de SAX-interface. De mogelijke datatypes in een Digital Item document zijn echter wel vrij uitgebreid en door de continue ontwikkeling van de MPEG-21 standaard kunnen deze nog toenemen in de toekomst. Omwille van deze redenen en om de parsing code afgescheiden van de datatypes te houden, is er een raamwerk ontwikkeld dat XML-elementen mapt op C++ klasses. Hiermee kunnen we met de SAX-interface onze specifieke boomstructuur opbouwen met minimale parsing code. In het raamwerk hebben we een klasse ontwikkeld waar de we de inhoud van een XML-document mee kunnen definiëren: `ContentDef`. In deze klasse wordt een definitie van elk datatype dat in het document kan voorkomen, bijgehouden. Een definitie van een datatype is gekoppeld aan een C++ klasse, en definieert welke attributen en kinderen het heeft en in welke member variabelen deze moeten worden opgeslagen. In Listing 4.1 wordt geïllustreerd hoe het raamwerk wordt gebruikt. Hier worden twee C++ klasses gegeven die het *Anchor*- en het *Component*-element uit de DIDL representeren. In de code daaronder voegen we de definities van de klassen toe aan een `ContentDef` klasse. De `ContentDef`-klasse houdt alle mogelijke datatypes bij die in een DIDL-document kunnen voorkomen, als we hier dan nog het wortelelement aan meegeven, hebben we de hiërarchie van datatypes die in het document kan

voorkomen. Met de `ElementDef`-klasses wordt de definitie van een datatype gegeven. We registeren hiermee welke attributen en kinderen een element kan hebben, en koppelen deze aan een member-variabele waarin deze in zullen worden opgeslagen. Als we een datatype in een `ElementDef`-klasse aan het parsen zijn en we komen een kind tegen dat geregistreerd is. Dan zoekt het parserraamwerk automatisch in de `ContentDef`-klasse naar de definitie van dit type en dan wordt hier mee verder gegaan. Als we alle datatypes die in een document kunnen voorkomen hebben gedefinieerd en we geven aan welk het wortelelement is, dan kunnen we met behulp van deze klasses automatisch onze hiërarchie van C++ klasses creëeren op basis van SAX-events.

Het voordeel van het gebruik van dit raamwerk is dat we onze eigen specifieke datastructuur direct kunnen creëeren van SAX-events met weinig extra code. Er is bovendien een losse koppeling tussen definities en klasses, waardoor de definities uit het geheugen kunnen verwijderd worden zodra het document is geparset. De enigste vereiste is dat de member-variabelen die de kinderen of attributen van een klasse bijhouden, afgeleid zijn van een abstracte containerklasse. De containerklasse voorziet de methodes om de geconstrueerde kinderen of de inhoud van een attribuut door te geven. Als we een document met de DOM-api naar onze eigen datastructuren zouden willen vertalen, zou er eerst nog een generieke boom geconstrueerd worden die relatief veel geheugen in beslag neemt. Deze stap is nu niet meer nodig. Het document hoeft ook niet in één keer ingeladen te worden, maar kan stapsgewijs geparset worden.

Om documenten in de Digital Item Declaration Language te parsen zijn er C++ klasses voor elke entiteit gecreëerd, en voor elk van deze klasses is een definitie gegeven. Het parserraamwerk kan nu de structuur van de documenten automatisch vertalen naar een interne boomstructuur voor zover de structuur van het document gedefinieerd is. Vanaf dat de structuur niet meer gedefinieerd is, kunnen de SAX-events doorgegeven worden aan een member-variabele van de klasse. In de Digital Item Declaration Language kan dit alleen in het *Resource*-, *Statement*- en *Fragment*-element gebeuren. Deze elementen kunnen willekeurige data bevatten, de andere elementen kunnen alleen elementen uit de DIDL bevatten en deze zijn gedefinieerd. De member-variabele bepaalt wat er met de inhoud van het element moet gebeuren. Er zijn enkele klasses gemaakt die met de inhoud van de elementen kunnen omgaan. Voor de *Resource*-klasse zal de inhoud van het element binaire data van een bron zijn, al dan niet geëncodeerd met base64. De klasse die deze inhoud verwerkt kan dan de data in het geheugen houden als deze niet te groot is, en anders eventueel wegschrijven naar een tijdelijk bestand. Voor een *Fragment*- en een *Statement*- klasse zal de data over het algemeen XML-data zijn of eventueel gewoon tekstuele data voor een *Statement*-klasse. Er is daarom een klasse ontwikkeld die eender welke `ContentDef`-klasse uit deze data kan herkennen, en deze data daar verder meer parsen. Via controller-klasses kan aan de DIDL-parser

```
class Anchor
{
public:
  Datatypes::Attribute<unsigned long> precedence;
  Datatypes::Attribute<XMLCh*>        id;
  Datatypes::ElementList<Condition>   conditions;
  Datatypes::ElementList<Descriptor>  descriptors;
  Datatypes::Element<Fragment>        fragment;
};

class Component
{
public:
  Datatypes::Attribute<XMLCh*>        id;
  Datatypes::ElementList<Condition>   conditions;
  Datatypes::ElementList<Descriptor>  descriptors;
  Datatypes::ElementList<Resource>    resources;
  Datatypes::ElementList<Anchor>      anchors;
};

class DIDLEngine
{
public:
  void createContentDefinition()
  {
    mDIDLDef
    [
      // ... other elements ...

      ElementDef<Anchor>()
      .attribute(s(ATTR_PRECEDENCE),        &Anchor::precedence),
      .attribute(s(ATTR_ID),                &Anchor::id)
      .child(s(NS_DIDL), s(ELT_CONDITION),  &Anchor::conditions)
      .child(s(NS_DIDL), s(ELT_DESCRIPTOR), &Anchor::descriptors)
      .child(s(NS_DIDL), s(ELT_FRAGMENT),   &Anchor::fragment),

      ElementDef<Component>()
      .attribute(s(ATTR_ID),                &Component::id),
      .child(s(NS_DIDL), s(ELT_CONDITION),  &Component::conditions),
      .chils(s(NS_DIDL), s(ELT_DESCRIPTOR), &Component::descriptors)
      .child(s(NS_DIDL), s(ELT_RESOURCE),   &Component::resources)
      .child(s(NS_DIDL), s(ELT_ANCHOR),     &Component::anchors),

      // ... other elements ...
    ];
  }

private:
  Parser::ContentDef<DIDL> mDIDLDef;
};
```

Listing 4.1: Voorbeeld van hoe C++ klasses aan elementdefinities gekoppeld woren.

meegegeven worden welke ContentDef-klasses er zich in een *Statement* of *Fragment* kunnen bevinden. Er is bijvoorbeeld een `ContentDef`-klasse voor het DIA-gedeelte ontwikkeld. Door deze contentdefinitie aan de DIDL-parser mee te geven kan de parser DIA-beschrijvingen in *Statement*-elementen herkennen en deze automatisch parsen.

## 4.2 Digital Item Adaptation

Het zevende deel van de MPEG-21 standaard, Digital Item Adaptation, standaardiseert verschillende tools die gebruikt kunnen worden bij de adaptatie van Digital Items. Er wordt echter wel niet gestandaardiseerd hoe deze tools met mekaar verbonden moeten worden en samen een adaptatie-engine kunnen vormen. Daarom is de functionaliteit van de tools in een library gestoken, zonder connecties tussen de verschillende tools te leggen. Dit zorgt ervoor dat applicaties die de library gebruiken zelf kunnen kiezen welke tools ze willen gebruiken en hoe ze ze met elkaar willen relateren. Er is een voorbeeld applicatie ontworpen die de tools gebruikt om een adaptatie-engine te construeren. De voorbeeld adaptatie-engine legt de connecties tussen de verschillende tools en voert de adaptatie uit.

### 4.2.1 Geïmplementeerde tools

De tools die geïmplementeerd zijn, zijn: BSDLink, Terminal and Network Quality of Service en BSD tools. Het zijn de tools die in deze thesis in detail zijn besproken. Deze tools worden binnen de Digital Item Adaptation standaard gebruikt om schaleerbare multimediabronnen te adapteren en Quality of Service (QoS) te beheren. Het parserraamwerk van de DIDL-parser wordt gebruikt om een definitie van alle geïmplementeerde DIA-datatypes te creëeren. Deze definitie wordt aan de DIDL-parser meegegeven. Als deze een dan een Digital Item gaat parsen en een DIA-beschrijving in een *Statement* tegenkomt, wordt deze automatisch vertaald naar de specifieke datatypes.

#### BSDLink

De implementatie van de BSDLink tool (zie Sectie 3.3) bestaat uit een definitie van de datatypes met behulp van het parserraamwerk dat in Sectie 4.1 beschreven is. De BSDLink-tool houdt alleen maar verwijzingen bij naar verschillende andere tools en heeft daarom geen functionaliteit op zijn eigen.

#### Terminal and Network Quality of Service

Deze implementatie bestaat uit een definitie van de AdaptationQoS beschrijving (zie Sectie 3.2) en de verschillende modules: UtilityFunction, LookUpTable en StackFunction. Ook

de werking van de verschillende modules is geïmplementeerd. De waarden van de input IOPins kunnen meegegeven worden en de AdaptationQoS-beschrijving kan hiermee de waarden van de output IOPins berekenen.

### Bitstream Syntax Description

De implementatie van de Bitstream Syntax Description tools (zie Sectie 3.1) bestaat uit twee grote delen. Eén deel is een BSDL-parser dat BS Schema's kan verwerken en deze kan gebruiken om BSD's van bitstreams te genereren en vice versa. Het andere deel is een geoptimaliseerd proces dat gBSD's naar bitstreams kan omzetten.

De BSDL-parser maakt extensief gebruikt van de Xerces-C++ bibliotheek en gedeeltelijk van de XQilla bibliotheek [XQI]. De Xerces-C++ bibliotheek voorziet onder andere een implementatie om XML-documenten te valideren tegen een XML Schema. De bibliotheek ondersteunt daarom de functionaliteit om een XML Schema naar een interne representatie om te zetten. De BSDL-parser gebruikt deze representatie om een eigen representatie van een BS Schema te creëeren. Deze eigen representatie is nodig omdat een BS Schema een heel andere functionaliteit biedt als een gewoon XML Schema, en omdat de diverse uitbreidingen nog geparset moet worden. De eigen representatie van een BS Schema implementeert de functionaliteit om datatypes naar een bitstream weg te schrijven en van een bitstream in te lezen. Met behulp van deze parser zijn er twee uitvoerbare processen gecreëerd. Het eerste proces kan een bitstream van een BSD genereren met behulp van een BS Schema. De implementatie gebruikt hiervoor een SAX-parser. Als de SAX-parser de start van een nieuw element doorgeeft dan volgen we ook in de boomstructuur van het BS Schema als dat nodig is. Als we bij een bladelement terecht komen dan gebruiken we de interne representatie van het datatype uit het BS Schema om de inhoud van dit element naar een bitstream weg te schrijven. Het tweede proces dat gebruik maakt van de BSDL-parser kan een BSD genereren van een bitstream en een BS Schema. De BSD wordt opgesteld met behulp van de DOM-api. Deze api houdt de BSD helemaal in het geheugen, dit is nodig omdat er XPath expressies uitgevoerd moeten worden over de boomstructuur. XPath expressies vereisen dat het hele document waarover de expressie uitgevoerd wordt, beschikbaar is. De XPath expressies worden uitgevoerd met behulp van de XQilla-bibliotheek. De DOM-api laat toe om een geconstrueerde boom naar een bestand uit te schrijven.

De gBSD's worden met behulp van een SAX-parser naar een bitstream omgezet. Het proces is hiervoor geöptimaliseerd in de zin dat het niet eerst een schema moet inladen. De betekenis van de verschillende elementen en de attributen die in een gBSD kunnen voorkomen liggen vast in de code.

### 4.2.2 Adaptatie-engine

De geïmplementeerde adaptatie-engine illustreert hoe de verschillende DIA-tools samengebracht kunnen worden om een adaptatie-engine te bouwen. De ontwikkelde adaptatie-engine kan gebruikt worden om digitale media in verschillende vormen op een transparante manier aan gebruikers met heterogene apparaten aan te bieden. Als de gebruiker een bepaald Digital Item wil consumeren dan stuurt hij of zij een een aanvraag hiervoor naar de server, samen met een beschrijving van zijn of haar omgeving. De geïmplementeerde adaptatie-engine gaat dan in het Digital Item kijken hoe dit afgestemd kan worden op de restricties van de omgeving van de gebruiker. Dit maakt de implementatie onafhankelijk van het systeem waarvoor geadapteerd wordt. De adaptatie-engine gaat in het Digital Item zoeken naar DIA-tools waarmee het de bronnen hierin kan adapteren. Met een *AdaptationQoS*-beschrijving gaat de adaptatie-engine de Quality of Service (QoS) voor de gebruiker proberen te maximaliseren. Hiermee gaan we berekenen hoe we de adaptatie optimaal kunnen uitvoeren om aan de restricties van de gebruiker te voldoen. De *Bitstream Syntax Description* tools maken de adaptatie-engine formaatonafhankelijk. In het Digital Item wordt verwacht dat er voor multimediabronnen die geadapteerd moeten worden, een Bitstream Syntax Description met bijbehorend BS Schema en transformatiestylesheet aanwezig is. Hierdoor hoeft de adaptatie-engine geen weet te hebben van de verschillende formaten van de multimediabronnen in het Digital Item. We zouden de BSD ook kunnen weglaten en deze genereren met het BS Schema als het DI geadapteerd is, maar aangezien deze voor iedere adaptatie nodig is lijkt het beter om deze op voorhand te genereren en als metadata toe te voegen. De *BSDLink*-tool voorziet de koppeling tussen de DIA-beschrijvingen die zojuist vernoemd zijn en de multimediabronnen.

De uitvoering van de adaptatie met behulp van de adaptatie-engine is gebaseerd op de aanwezige DIA-tools in het Digital Item en een beschrijving van de omgeving van de gebruiker. De auteur van het Digital Item heeft dus de volledige controle over welke adaptaties zijn toegestaan en welke adaptaties uitgevoerd worden om aan de restricties van de gebruiker te voldoen. Het is niet altijd gewenst voor een auteur dat een gebruiker alle mogelijke adaptaties van een bron kan verkrijgen. Een kunstenaar bijvoorbeeld kan alleen toelaten dat zijn werk in bepaalde hoge resoluties te verkrijgen is. Een andere mogelijkheid is bijvoorbeeld dat we één binaire bron voor een video in hoge kwaliteit bewaren, maar hier in een Digital Item twee Componenten voor creëeren. We zouden voor één Component dan alleen adaptaties met een lage kwaliteit kunnen toelaten, en voor de andere alle mogelijke adaptaties. Dit systeem zou een uitgever kunnen gebruiken om verschillende versies van één bron tegen verschillende tarieven aan te bieden zonder deze dubbel te moeten bewaren.

De geïmplementeerde adaptatie-engine zou ingezet kunnen worden in een scenario waar een multimedia server verschillende digitale media publiceert voor een heterogene groep van ge-

bruikers. Het systeem zou niet alleen kosten voor de server kunnen besparen door een gereduceerd bandbreedteverbruik, maar het zou bovendien ook nog de Quality of Service voor de gebruiker verbeteren.

### Verloop van de adaptatie

De adaptate-engine illustreert hoe de geïmplementeerde tools samengebracht kunnen worden om een adaptatie-engine te bouwen. De connecties tussen de verschillende tools worden gelegd en de functionaliteit van elke tool wordt uitgebuit. Het proces is in geen zin gestandaardiseerd, het is de bewuste keuze van MPEG om dit vrij te laten aan de implementatie. Dit is gedaan om gezonde concurrentie te promoten in de hoop naar de toekomst toe steeds betere implementaties te bekomen. Hieronder worden de verschillende fases besproken die de geïmplementeerde adaptatie-engine doorgaat om een Digital Item te adapteren aan de omgeving van de gebruiker:

1. In de eerste fase krijgt de adaptatie-engine een aanvraag van een gebruiker binnen voor een Digital Item. Bij deze aanvraag stuurt de gebruiker een Digital Item mee dat de omgeving van de gebruiker beschrijft. Voor deze beschrijving worden de *Usage Environment Description*-tools gebruikt. De gebruiker kan hiermee bijvoorbeeld aangeven wat zijn beschikbare schermgrootte is, wat de bandbreedte van het netwerk is, ... etc. Omdat we nu met twee Digital Items hebben te maken zal in de komende tekst naar het Digital Item dat aangevraagd wordt, gerefereerd worden als het *Content-DI* en naar het Digital Item met de beschrijving van de gebruiker als het *Context-DI*. Als respons op de aanvraag gaat de adaptatie-engine het Content-DI parsen met behulp van de DIDL-parser die in de vorige sectie is besproken.

2. Vervolgens gaan we de interne representatie van het Content-DI overlopen om alle *Component*-elementen terug te vinden. De functie van deze elementen is om tekstuele informatie aan multimediabronnen te koppelen. Zulke informatie kan bijvoorbeeld een DIA-beschrijving zijn.

3. Voor elke Component-element dat we hebben gelocaliseerd, gaan we op zoek naar een *BSDLink*-beschrijving die geässocieerd is met de multimediabron binnen het element. De BSDLink-beschrijving creëert de adaptatie-architectuur voor de Component, de beschrijving zal ons helpen de andere nodige DIA-tools te localiseren en de links te leggen tussen hen.

4. De eerste tool die we gaan uitbuiten waar de BSDLink-beschrijving ons naar verwijst, is een *AdaptationQoS*-beschrijving. Deze verwijzing kan naar een *Statement* binnen het huidige document zijn of naar een DIA-beschrijving in een ander document. Als we met het laatste geval hebben te maken moet deze eerst nog geparset worden.

5. Eens de AdaptationQoS-beschrijving geparset en beschikbaar is, worden de waarden van de input IOPins ingevuld. Met behulp van het *semantics*-attribuut kunnen we een semantische betekenis geven aan een IOPin. Afhankelijk van deze waarde gaan we een waarde uit de UED-beschrijving in het Context-DI halen en deze als waarde aan de IOPin geven.

6. Als de AdaptationQoS-beschrijving uitgevoerd is, hebben we de optimale transformatieparameters berekend voor de adaptatie zodat het resultaat nog aan de restricties van de gebruiker voldoet. In deze fase gebruiken we de Xalan-C++ [XAL] bibliotheek om een XSLT-transformatie uit te voeren op de BSD van de multimediabron met als input de outputwaarden berekend door de AdaptationQoS beschrijving. We gebruiken de BSDLink-beschrijving om te weten welke AdaptationQoS outputwaarden we als inputwaarden voor de transformatie moeten gebruiken. De verwijzing naar de BSD en de stylesheet halen we uit de BSDLink-beschrijving.

7. Als de BSD of gBSD getransformeerd is, gaan we deze terug omzetten naar een binaire bron. Als we een BSD naar een binaire bron willen omzetten hebben we het BS Schema nodig dat gebruikt is om de BSD te genereren. Een verwijzing naar het schema halen we uit de BSD.

8. Als elk Component-element geadapteerd is aan de omgeving van de gebruiker gaan we een nieuwe instantie van het Digital Item naar de gebruiker sturen waarbij elk Component-element naar de geadapteerde bron verwijst.

# Hoofdstuk 5

# Resultaten

De resultaten die de adaptatie-engine kan realiseren zijn sterk afhankelijk van de invoer die gegeven wordt. Bitstreams worden geadapteerd aan de hand van een beschrijving van het formaat, een beschrijving van het doel-systeem en een beschrijving die deze twee aan elkaar relateert. Hoe bitstreams geadapteerd kunnen worden is afhankelijk van de schaleerbaarheid van het formaat. In de komende secties worden enkele resultaten besproken die gegenereerd zijn met de geïmplementeerde adaptatie-engine. In de eerste sectie wordt besproken wat we kunnen realiseren door JPEG2000-afbeeldingen aan de omgeving van de gebruiker aan te passen. Voor elk resultaat wordt de invoer die de gebruiker aan het systeem geeft, de originele multimediabron en de nieuwe multimediabron die de adaptatie-engine gecreëerd heeft, getoond. In de tweede sectie wordt besproken hoe we een MPEG-4 videostroom kunnen adapteren. Het bestandsformaat van deze videostromen laat alleen maar toe een videostroom te adapteren in het temporele domein. Het effect van de adaptatie op de resulterende bitstream is hierdoor moeilijk te tonen in deze thesis, daarom is er gekozen om in deze sectie het effect van de adaptatie op de BSD of gBSD te tonen. De mogelijkheden van de geïmplementeerde adaptatie-engine zijn sterk afhankelijk van de schaleerbaarheid van het formaat van de multimediabronnen, maar de adaptatie-engine is wel onafhankelijk van het formaat van de multimediabronnen. Dit zorgt ervoor dat als er in de toekomst nieuwe bestandsformaten ontwikkeld zouden worden die betere schaleerbaarheids-opties bieden, de adaptatie-engine hiervoor niet aangepast moet worden, maar wel meer adaptatie-mogelijkheden kan bieden. Het enigste wat dan vereist is, is dat er een BS Schema ontwikkeld wordt dat het nieuwe formaat beschrijft, en een stylesheet die BSD's voor zulke bitstreams kan transformeren.

## 5.1 Adapteren van JPEG2000-afbeeldingen

De recent ontwikkelde JPEG2000-standaard is een schaleerbaar coderingsalgoritme voor afbeeldingen [ISO00]. De standaard biedt verscheidene manieren om afbeeldingen te adapteren

door simpele bitstreamoperaties zoals kleine wijzigingen in de bitstream en het weglaten van datasegmenten. JPEG2000-afbeeldingen zijn schaleerbaar in de volgende domeinen [ISO00]: het aantal kleurlagen kan gewijzigd worden van drie naar één (grijswaarden); de spatiale resolutie kan gereduceerd worden; en de afbeelding is opgebouwd uit een een aantal kwaliteitslagen die weggelaten kunnen worden, het aantal kwaliteitslagen bepaalt het Signal-to-Noise Ratio (SNR) van de geadapteerde afbeelding. Hieronder volgt de bespreking van enkele resultaten die bekomen zijn door de geïmplementeerde adaptatie-engine.

### 5.1.1 Bandbreedte en schermgrootte

In Figuur 5.1(c) wordt een JPEG2000-afbeelding getoond die door de adaptatie-engine geadapteerd is. Dit resultaat is bekomen door de originele afbeelding, links in de figuur, aan de beschikbare bandbreedte en de schermgrootte van de gebruiker aan te passen. Het originele Digital Item bestaat uit de afbeelding links in de figuur en enkele DIA-beschrijvingen die bepalen hoe de afbeelding geadapteerd kan worden. De adaptatie is uitgevoerd op basis van een UED-beschrijving die de schermgrootte van de gebruiker en de gemiddelde beschikbare bandbreedte beschrijft. De AdaptationQoS-beschrijving die in het originele Digital Item werd gebruikt is een combinatie van de drie voorbeelden uit Sectie 3.2 en nog een extra StackFunction-module. De StackFunction-module vermenigvuldigt de bandbreedte die we als invoer krijgen met een constante. Deze constante bepaalt de maximale tijd die we willen toelaten voor de transmissie van de afbeelding. De gecombineerde AdaptationQoS-beschrijving garandeert een maximum tijd dat de gebruiker moet wachten op een afbeelding en garandeert dat de afbeelding op zijn scherm past.

In Listing 5.1(a) is de UED-beschrijving getoond die het systeem en het netwerk van de gebruiker beschrijft. De gebruiker stuurt deze mee als hij een aanvraag verstuurd voor het Digital Item met de foto. In de beschrijving staat dat de resolutie van het scherm van de gebruiker $400 \times 300$ pixels is en dat de de gemiddelde beschikbare bandbreedte 560000 bits per seconde is. Stel dat we een transmissietijd van 1 seconde willen bereiken, met een StackFunction-module berekent de adaptatie-engine dat we in deze tijd 70kB over het netwerk kunnen verzenden. De UtilityFunction-module uit Listing 3.9 berekent uit deze waarde dat de spatiale resolutie van de afbeelding gereduceerd moet worden tot een kwart en dat we maar drie kwaliteitslagen van de afbeelding mogen overhouden. De LookUpTable-module uit Listing 3.10 berekent uit de resolutie van de gebruiker dezelfde reductie in spatiale resolutie als de vorige module. De functie van de StackFunction-module uit Listing 3.11 is om de uitvoerwaarden van de twee modules te combineren, maar beiden resulteren in dezelfde waarde voor schalering in spatiale resolutie. Als de twee modules een verschillende waarden berekenen, dan gaat de StackFunction-module uit Listing 3.11 de parameter kieze die zorgt voor de grootste reductie

```
<UsageEnvironment xsi:type="TerminalCapabilitiesType">
  <TerminalCapabilities xsi:type="InputOutputCapabilitiesType">
    <Display>
      <Resolution horizontal="400" vertical="300"/>
    </Display>
  </TerminalCapabilities>
</UsageEnvironment>
<UsageEnvironment xsi:type="NetworkCharacteristicsType">
  <NetworkCharacteristics xsi:type="NetworkConditionType">
    <AvailableBandwidth average="560000" />
  </NetworkCharacteristics>
</UsageEnvironment>
```

(a) UED-beschrijving van het netwerk en het systeem van de gebruiker

(b) Originele afbeelding

(c) Geadapteerde afbeelding

Figuur 5.1: Adaptatie aan de hand van de beschikbaare bandbreedte en schermgrootte.

in spatiale resolutie. Dit zorgt ervoor dat de adaptatie aan beide restricties voldoet.

Als de adaptatie-engine de uitvoerwaarden van de AdaptationQoS-beschrijving heeft berekend kan de transformatie van de BSD of de gBSD worden uitgevoerd. Vervolgens wordt een nieuwe bitstream van de BSD of de gBSD gegenereerd. Het resultaat dat we bekomen na de bitstreamadaptatie is in Figuur 5.1(c) te zien. Het is de originele afbeelding waarbij de spatiale resolutie geschaleerd is tot een kwart, en waarbij er drie kwaliteitslagen uit de originele afbeelding met vijf lagen zijn overgehouden. De gebruiker ontvangt een Digital Item dat naar de geadapteerde afbeelding verwijst in plaats van naar de originele afbeelding.

### 5.1.2 Kleurenblindheid

We kunnen de adaptatie-engine niet alleen gebruiken om multimediabronnen aan de restricties van het systeem van de gebruiker aan te passen, maar ook om bronnen aan bijvoorbeeld een handicap van de gebruiker aan te passen. In het volgende resultaat is te zien hoe dat we de adaptatie-engine rekening kunnen laten houden met een visuele handicap van de gebruiker. De UED-beschrijving uit Listing 5.2(a) geeft informatie over de graad van kleurenblindheid dat de gebruiker heeft. De visuele handicap kan door drie graden worden beschreven: *Severe*, *Medium* en *Mild*. Het originele Digital Item bevat de afbeelding uit Figuur 5.2(b). De afbeelding is geässocieerd met een AdaptationQoS-beschrijving die de verschillende tekstuele waarden voor graden met een aantal kleurlagen associeert. Als de gebruiker uit dit voorbeeld een aanvraag voor het Digital Item met de afbeelding verzendt, dan stuurt hij de UED uit Listing 5.2(a) mee. Deze beschrijft zijn kleurenblindheid als *Severe*. De adaptatie-engine gaat de AdaptationQoS-beschrijving uit het Digital Item uitvoeren en bepaalt hiermee dat er voor deze handicap slechts één kleurlaag moet doorgestuurd worden. Vervolgens wordt er een nieuwe afbeelding gegenereerd door een transformatie uit te voeren op de BSD of gBSD uit het Digital Item met deze parameter. De geadapteerde afbeelding die wordt doorgestuurd is in Figuur 5.2(c) te zien. De gebruiker ontvangt een Digital Item dat naar de geadapteerde afbeelding verwijst in plaats van naar de originele.

### 5.1.3 Region Of Interest

Als de encodering van een JPEG2000-afbeelding het toelaat kunnen we een JPEG2000-afbeelding ook adapteren door er een rechthoekige regio uit te selecteren en alleen deze over te houden. Een JPEG2000-afbeelding kan zo geëncodeerd worden dat de afbeelding in een rooster is opgedeeld. We kunnen dan een een rechtoekige regio uit de afbeelding selecteren en alles wat hier buiten valt weglaten door simpelweg datasegmenten uit de JPEG2000-afbeelding weg te laten. De hoekpunten van de rechthoekige regio moeten wel samenvallen met punten op het rooster. Als de resolutie van het rooster heel groot is, zullen we ook heel specifieke regio's kunnen selecteren, maar dan zal dit wel voor een grotere bitstream zorgen [ISO00].

Stel dat de gebruiker een Digital Item wil bekijken dat een afbeelding met een zeer hoge resolutie bevat en hij wil hier een specifiek punt in zoeken. We kunnen de gebruiker dan eerst een lage resolutie van de afbeelding sturen. Hier kan hij dan een rechthoekige regio in selecteren. Vervolgens sturen we deze regio in een hogere resolutie naar de gebruiker. Met deze methode kan de gebruiker stapsgewijs een punt zoeken zonder dat we hem de hele afbeelding in een hoge resolutie moeten sturen.

In Listing 5.3(a) wordt de UED-beschrijving getoond die we hiervoor als invoer aan de

```
<UsageEnvironment xsi:type="UserCharacteristicsType">
  <UserCharacteristics xsi:type="AccessibilityCharacteristicsType">
    <Visual>
      <ColorVisionDeficiency>
        <DeficiencyType>CompleteColorBlindness</DeficiencyType>
        <DeficiencyDegree>
          <TextualDegree>Severe</TextualDegree>
        </DeficiencyDegree>
      </ColorVisionDeficiency>
    </Visual>
  </UserCharacteristics>
</UsageEnvironment>
```

(a) UED-beschrijving van de visuele handicap van de gebruiker

(b) Originele afbeelding

(c) Geadapteerde afbeelding

Figuur 5.2: Adaptatie aan de hand van kleurenblindheid.

adaptatie-engine kunnen geven. De UED-beschrijving definieert een *Focus Of Attention* van de gebruiker. Het is een link naar een MPEG-7 beschrijving die een regio binnen de afbeelding selecteert waar de gebruiker in geïnteresseerd is. De MPEG-7 beschrijving die de rechthoek definieert is te zien in Listing 5.3(b). Deze beschrijving staat apart omdat de UED-tool ons alleen toe laat een URI naar een MPEG-7 beschrijving te geven voor het *ROI*-element en niet om de data in het element op te nemen. Als we deze beschrijving van de gebruiker ontvangen hebben we alleen nog maar een AdaptationQoS-beschrijving nodig die de geselecteerde regio mapt op de dichtsbijzijnde roosterpunten op de afbeelding. Als we deze roosterpunten hebben kunnen we de regio met behulp van een stylesheet uit de afbeelding halen, en een nieuwe bitstream genereren die alleen deze regio bevat. In Figuur 5.3(c) is de originele afbeelding in een lage resolutie afgebeeld. In Figuur 5.3(d) is de geädapteerde afbeelding afgebeeld. Deze afbeelding bevat alleen de regio die in de MPEG-7 beschrijving in Listing 5.3(b) is gedefinieerd. De geselecteerde regio is ook twee keer zo groot als de regio in Figuur 5.3(c).

## 5.2 Adapteren van MPEG-4 Visual Elementary Streams

De adaptatie-engine beperkt zich niet tot het adapteren van afbeeldingen. De adaptatie-engine is zelfs niet bewust van de vorm van multimedia die het adapteert. In deze sectie wordt besproken hoe dat we videostromen in het MPEG-4 Visual Elementary Stream (VES) [ISO94b] formaat kunnen adapteren met behulp van de adaptatie-engine. MPEG-4 VESs zijn videostromen die in een MPEG-4 container kunnen voorkomen. De bitstreams bevatten alleen de grafische beelden die een video vormen en geen audio data. Het formaat is niet ontworpen met schaleerbaarheid in gedachte maar desondanks kunnen de videostromen toch op een beperkte manier geadapteerd worden door datasegmenten weg te laten uit de bitstream.

### 5.2.1 Temporele schaleerbaarheid

Een MPEG-4 Visual Elementary Stream (VES) is een opeenvolging van visuele frames. Elk frame is geässocieerd met een tijdstip in het videosegment en kan tot het volledige beeld dat hiermee overeenkomt, gedecodeerd worden. De binaire data waar een frame uit bestaat, valt in één continu segment in de bitstream en is niet gemixed tussen data van andere frames. Hierdoor kunnen we met de BSD tools hele frames uit de bitstream weglaten zonder dat de data van andere frames beïnvloed wordt. Een frame bevat echter wel niet noodzakkelijk alle informatie om een beeld te construeren. Er zijn drie soorten frames [ISO94b]:

- Een I-frame bevat alle informatie om een beeld te construeren. Het decoderen ervan is niet afhankelijk van andere frames.
- Een P-frame bevat de verschillen met het vorige I- of P-frame. Het decoderen ervan is afhankelijk van het vorige gedecodeerde I- of P-frame.
- Een B-frame bevat de verschillen met het vorige en het volgende frame I- of P-frame. Het decoderen is afhankelijk van het vorige en het volgende gedecodeerde I- of P-frame.

De adaptatie-engine kan een MPEG-4 Visual Elementary Stream adapteren door deze te schaleren op de temporele as. Er kunnen bepaalde type frames verwijderd worden. Het decoderen van sommige frames is wel afhankelijk van anderen, hier moet dus rekening mee gehouden worden. Het resultaat van deze bewerking is dat de bestandsgrootte van het videosegment gereduceerd wordt door de framerate te verkleinen. Omdat een verkleinde framerate van een video moeilijk te tonen is, wordt in deze tekst het effect van de adaptatie op de gBSD gedemonstreerd.

In Listing 5.4(a) wordt de gBSD van een videostroom uit een Digital Item getoond. In het begin van de gBSD zien we enkele gBSDUnits die delen van de header van een MPEG-4 videostroom beschrijven. Daar achter volgen de gBSDUnits die de opeenvolgende frames in

```
<xsl:stylesheet xmlns="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
                xmlns:gbsd="urn:mpeg:mpeg21:2003:01-DIA-gBSD-NS"
                xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0" >
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:param name="removeLabel" select="':M4V:B_VOP'"/>
  <!-- Match all: default template -->
  <xsl:template name="tplAll" match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- Match gBSDUnit: removes VOPs -->
  <xsl:template name="tplgBSDUnitLabel"
                match="gbsd:gBSDUnit[@syntacticalLabel]">
    <xsl:if test="@syntacticalLabel!=\$removeLabel">
      <xsl:copy>
        <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Listing 5.1: XSLT-stylesheet dat B-frames weglaat [ISO08]

de videostroom beschrijven. De frames dragen het *syntacticalLabel*-attribuut met als waarde `:M4V:I_VOP`, `:M4V:P_VOP` of `:M4V:B_VOP` dat respectievelijk een I-, een P- of een B-frame aanduidt. Als een gebruiker vereist dat de bestandsgrootte van de videostroom kleiner is, kan de adaptatie-engine de gBSD transformeren met behulp van een stylesheet dat bijvoorbeeld alle B-frames weglaat. In Listing 5.4(b) is de gBSD te zien die de adaptatie-engine bekomt na deze transformatie. Als de adaptatie-engine een nieuwe bitstream van deze gBSD genereert krijgen we een nieuwe MPEG-4 VES met een lagere framerate en een lagere bestandsgrootte. In Listing 5.1 is een XSLT-stylesheet te zien dat een bepaalt soort frames kan weglaten. Het kopieert alle elementen uit het brondocument naar het doeldocument, behalve voor gBSDUnits wordt het syntacticalLabel-attribuut gecontroleerd.

### 5.2.2 Semantische adaptatie

De geïmplementeerde adaptatie-engine kan ook ingezet worden om bitstreams op basis van semantische informatie te adapteren. Het gBS Schema laat toe om in een gBSD semantische informatie te geven over bepaalde datasegmenten door middel van het *marker*-attribuut dat aan het *Parameter*- of het *gBSDUnit*-element kan meegegeven worden. In een gewone BSD kunnen we zelf kiezen hoe we semantische informatie willen toevoegen want we kunnen onze eigen specifieke datatypes in het bijbehorend BS Schema definiëren. Een transformatiesty-

lesheet kan dan rekening houden met deze informatie als de BSD of gBSD getransformeerd wordt.

Stel nu dat een uitgeverij een nieuwe film wilt publiceren. De film bevat verschillende geweldadige scènes die niet geschikt zijn voor jongere kijkers. In plaats van de hele film voor jongeren te verbieden, kiest de uitgeverij ervoor om de film dynamisch te adapteren aan de leeftijd van de consument. Afhankelijk van de leeftijd van de consument wil de uitgeverij dat bepaalde scènes verwijderd worden. Dit concept kan met de geïmplementeerde adaptatie-engine gerealiseerd worden. We vertrekken met de film in het MPEG-4 VES formaat en een MPEG-7 beschrijving die semantische informatie over de film verschaft. De MPEG-7 beschrijving is een temporele decompositie van de film die aanduidt welke scènes geweldadig zijn en hoe geweldadig ze zijn. Met behulp van de geïmplementeerde BSD tools en een BS Schema voor MPEG-4 VES genereren we een BSD uit de bitstream van de film. Een XSLT-stylesheet transformeert de BSD naar een gBSD en gebruikt de MPEG-7 beschrijving om de semantische informatie aan de gBSD toe te voegen in de *marker*-attributen. De semantisch informatie kan even goed aan de BSD worden toegevoegd, maar aangezien er een stylesheet beschikbaar is dat een gBSD creëert met deze informatie wordt dit gebruikt. De uitgeverij kan een Digital Item creëeren dat de geconstrueerde gBSD, een transformatiestylesheet dat bewust is van de waarden in de *marker*-attributen en een AdaptationQoS-beschrijving die leeftijden associeert met toegelaten geweldadigheidsniveau's bevat. De adaptatie-engine kan de film dan op een transparante manier adapteren voor de gebruiker.

In Listing 5.5(a) wordt een uittreksel van de gBSD getoond die de film beschrijft. De gBSD is aangevuld met semantische informatie. Alle frames uit geweldadige scènes in de film zijn gegroepeerd in gBSDUnits waarbij het *marker*-attribuut een graad van geweldadigheid aanduidt. De waarde die het attribuut kan hebben is in de vorm "`violent-x`", waarbij $x$ een numerieke waarde is die de graad aanduidt. Het transformatiestylesheet dat in het Digital Item geässocieerd is met de gBSD laat toe om deze scènes te verwijderen. Het stylesheet krijgt een numerieke parameter mee, en zal alle scènes verwijderen waarbij $x$ groter is als de parameter. Een AdaptationQoS-beschrijving in het Digital Item associeert leeftijden met een toegelaten graad van geweldadigheid. Als de adaptatie-engine een aanvraag voor het Digital Item krijgt samen met een UED-beschrijving die de leeftijd van de gebruiker geeft, creëert de adaptatie-engine en nieuw Digital Item met hierin de geadapteerde film. Deze geadapteerde film zal alleen de scènes bevatten die de uitgeverij gepast acht voor de leeftijd van de gebruiker. In Listing 5.5(b) wordt de geadapteerde gBSD getoond. Hier is de scène met een geweldadigheidsniveau dat groter is als twee verwijderd.

```
<UsageEnvironment xsi:type="UserCharacteristicsType">
  <UserCharacteristics xsi:type="PresentationPreferencesType">
    <FocusOfAttention>
      <ROI uri="stillroi.xml#region1"/>
    </FocusOfAttention>
  </UserCharacteristics>
</UsageEnvironment>
```

(a) UED-beschrijving van de geselecteerde regio

```
<Mpeg7>
  <DescriptionUnit xsi:type="StillRegionType" id="region1">
    <SpatialMask>
      <SubRegion>
        <Box mpeg7:dim="2 2">0 128 480 384</Box>
      </SubRegion>
    </SpatialMask>
  </DescriptionUnit>
</Mpeg7>
```

(b) MPEG-7-beschrijving die de geselecteerde regio beschrijft

(c) Originele afbeelding

(d) Geadapteerde afbeelding

Figuur 5.3: Adaptatie aan de hand van een geselecteerde Region Of Interest (ROI)

```
<Header>
  <DefaultValues addressUnit="byte" addressMode="Absolute" />
  <ClassificationSchemeAlias alias="M4V"
      href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
</Header>
<!-- start video object sequence -->
<gBSDUnit syntacticalLabel=":M4V:VOS" start="0" length="10"/>
<!-- start video object -->
<gBSDUnit syntacticalLabel=":M4V:VO" start="10" length="4"/>
<!-- start video object layer -->
<gBSDUnit syntacticalLabel=":M4V:VOL" start="14" length="28"/>
<!-- video object planes (frames) -->
<gBSDUnit syntacticalLabel=":M4V:I_VOP" start="42" length="46749"/>
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="46791" length="393"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="47184" length="70"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="47254" length="71"/>
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="47325" length="389"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="47714" length="56"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="47770" length="56"/>
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="47826" length="351"/>
<!-- ... en zo verder ... -->
```

(a) Originele gBSD

```
<Header>
  <DefaultValues addressUnit="byte" addressMode="Absolute" />
  <ClassificationSchemeAlias alias="M4V"
      href="urn:mpeg:mpeg4:visual:cs:syntacticalLabels"/>
</Header>
<!-- start video object sequence -->
<gBSDUnit syntacticalLabel=":M4V:VOS" start="0" length="10"/>
<!-- start video object -->
<gBSDUnit syntacticalLabel=":M4V:VO" start="10" length="4"/>
<!-- start video object layer -->
<gBSDUnit syntacticalLabel=":M4V:VOL" start="14" length="28"/>
<!-- video object planes (frames) -->
<gBSDUnit syntacticalLabel=":M4V:I_VOP" start="42" length="46749"/>
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="46791" length="393"/>
<!-- B-frame verwijderd -->
<!-- B-frame verwijderd -->
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="47325" length="389"/>
<!-- B-frame verwijderd -->
<!-- B-frame verwijderd -->
<gBSDUnit syntacticalLabel=":M4V:P_VOP" start="47826" length="351"/>
<!-- ... en zo verder ... -->
```

(b) Getransformeerde gBSD

Figuur 5.4: Adaptatie van een MPEG-4 VES door het weglaten van B-frames

```
<!-- ... frames ... -->
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="95792" length="69"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="95861" length="69"/>
<gBSDUnit marker="violent-2" start="95930" length="24939295">
  <gBSDUnit syntacticalLabel=":M4V:P_VOP"
            start="95930" length="389"/>
  <!-- ... frames ... -->
  <gBSDUnit syntacticalLabel=":M4V:B_VOP"
            start="25034257" length="968"/>
</gBSDUnit>
<gBSDUnit syntacticalLabel=":M4V:I_VOP"
          start="25035225" length="68812"/>
<!-- ... frames ... -->
<gBSDUnit syntacticalLabel=":M4V:P_VOP"
          start="43822808" length="16470"/>
<gBSDUnit marker="violent-3" start="43839278" length="8701470">
  <gBSDUnit syntacticalLabel=":M4V:I_VOP"
            start="43839278" length="16640"/>
  <!-- ... frames ... -->
  <gBSDUnit syntacticalLabel=":M4V:B_VOP"
            start="52530148" length="10600"/>
</gBSDUnit>
<gBSDUnit syntacticalLabel=":M4V:I_VOP"
          start="52540748" length="25671"/>
<!-- ... frames ... -->
```

(a) Originele gBSD

```
<!-- ... frames ... -->
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="95792" length="69"/>
<gBSDUnit syntacticalLabel=":M4V:B_VOP" start="95861" length="69"/>
<gBSDUnit marker="violent-2" start="95930" length="24939295">
  <gBSDUnit syntacticalLabel=":M4V:P_VOP"
            start="95930" length="389"/>
  <!-- ... frames ... -->
  <gBSDUnit syntacticalLabel=":M4V:B_VOP"
            start="25034257" length="968"/>
</gBSDUnit>
<gBSDUnit syntacticalLabel=":M4V:I_VOP"
          start="25035225" length="68812"/>
<!-- ... frames ... -->
<gBSDUnit syntacticalLabel=":M4V:P_VOP"
          start="43822808" length="16470"/>

<!-- scene weggelaten door transformatiestylesheet -->

<gBSDUnit syntacticalLabel=":M4V:I_VOP"
          start="52540748" length="25671"/>
<!-- ... frames ... -->
```

(b) Geadapteerde gBSD

Figuur 5.5: Adaptatie van MPEG-4 VES op basis van semantische informatie

# Hoofdstuk 6

# MPEG-21 in de praktijk

Verschillende delen van de MPEG-21 standaard zijn reeds enkele jaren afgwerkt, waaronder het Digital Item Adaptation gedeelte. Het is dan ook niet verwonderlijk dat enkele bedrijven reeds gebruik willen maken van deze nieuwe technologieën. Doordat de afgewerkte delen nog maar sinds enkele jaren gestandaardiseerd zijn is de standaard uiteraard nog niet wereldwijd geadopteerd. In deze sectie worden enkele bedrijven en Europese projecten besproken die applicaties hebben ontwikkeld rond de MPEG-21 standaard.

## 6.1 Enikos

Enikos is het eerste spin-off bedrijf van de *University of Wollongon* [ENI]. Het bedrijf is in 2003 opgestart met als doelstelling het onderzoek naar MPEG-21 binnen de universiteit te commercialiseren. De universiteit heeft al jaren onderzoek naar de standaard gedaan en is zelf ook betrokken bij de ontwikkeling van de standaard.

Het bedrijf heeft het *Enhanced Media Platform* ontwikkeld, een dynamisch en formaatonafhankelijk advertentieplatform [ENI]. Het platform laat uitgevers toe om met een Web 2.0 applicatie door simpele drag-and-drop operaties interactieve advertenties samen te stellen. In deze advertenties kunnen verschillende vormen van multimedia opgenomen worden. Enikos hoopt hiermee advertenties doeltreffender te maken. De advertenties die door het platform geconstrueerd worden zijn Digital Items die voldoen aan de MPEG-21 standaard. Welke delen van de standaard en welke tools hieruit ze voor hun platform gebruiken, vermelden ze niet.

## 6.2 Adactus

Adactus is een Noors bedrijf dat een nieuw platform voorziet waarmee gebruikers op een transparante manier multimedia op mobiele apparatuur kunnen consumeren [ADA]. Om

transparantie bij de adaptatie van de multimediabronnen te bekomen, wordt er gebruik gemaakt van delen uit de Digital Item Adaptation standaard. Adactus is vooral gericht op broadcasting scenario's. Adactus heeft applicaties ontwikkeld waarmee uitgevers of auteurs hun creaties op een multimedia server beschikbaar kunnen maken. Via een live-feed kunnen gebruikers bewust worden van deze nieuwe toevoegingen, en kunnen ze deze op een transparante manier van op bijna elk apparaat consumeren. De applicaties maken gebruik van MPEG-21 om de specificaties van het apparaat en het netwerk van de gebruiker te weten te komen, en sturen een Digital Item naar de gebruiker dat verwijst naar multimediabronnen die specifiek voor het apparaat zijn geädapteerd. Adactus laat ook toe om delen van Digital Items, zoals bijvoorbeeld advertenties, af te stemmen op de karakteristieken van de gebruiker.

## 6.3 DANAE

DANAE (**D**ynamic distributed **A**daptation of scalable multimedia co**N**tent in a context-**A**ware **E**nvironment) is een conglomeraat van een elftal partners. Het is een Europees gesponsord project dat als doel heeft de dynamische en gedistribueerde adaptatie van schaleerbare media in een gebruikers-bewuste omgeving te realiseren [DAN]. Het project is in 2004 van start gegaan en is in 2006 succesvol ten einde gebracht.

De architectuur die door het DANAE-project is ontwikkeld, is gebaseerd op de MPEG-21 standaard. Het *Digital Item Declaration* deel wordt gebruikt om multimedia in een Digital Item container te plaatsen. Onder andere *Digital Item Adaptation* tools worden gebruikt voor de transparante adaptatie van multimedia. De *Digital Item Processing (DIP)* tools worden gebruikt om het Digital Item dynamisch te maken door het proces te sturen. De DIP tools kunnen gezien worden als een functionele abstractielaag tussen het Digital Item en de applicatie. Ze voorzien een aantal abstracte methodes die door het Digital Item aangeroepen kunnen worden, en de applicatie implementeert deze methodes.

Het DANAE-project heeft drie aanpakken onderzocht om Digital Items te adapteren:

- **Static stream selection** creëert een aantal geädapteerde versies van multimediabronnen tijdens de creatie van het Digital Item. In het Digital Item wordt een referentie naar de verschillende versies opgenomen. Door het Digital Item te configureren wordt bepaald welke versies aan de gebruiker worden aangeboden. De DIP tools worden gebruikt om de configuratie van het Digital Item te sturen.
- **Dynamic stream selection** wordt gerealiseerd door de bitstreams van multimediabronnen op te delen in verschillende fragmenten. Elk fragment wordt gegroepeerd met de metadata die het deel beschrijft en de metadata die helpt bij het nemen van adaptatiebeslissingen voor het fragment. Voor elk fragment kan de adaptatie-engine een nieuwe

adaptatie-beslissing nemen. De adaptatie-engine van DANAE implementeert het adapteren van bronnen met behulp van de BSD tools en door middel van transcoderen. De transcodeermethode is zeer CPU-intensief en wordt alleen gebruikt als adaptatie met behulp van de BSD tools niet mogelijk is of niet geschikt is.

- **Distributed adaptation** laat toe om de adaptatie bij elke gebruiker uit te voeren. Een gebruiker kan bijvoorbeeld de multimediaserver zijn, een proxyserver of de consument. Door de fragmenten die in vorig punt besproken zijn samen met hun metadata tools te versturen kan elke gebruiker elk fragment apart adapteren.

## 6.4 Enthrone

Enthrone is net als DANAE een Europees gesponsord project. Het project heeft als doel een architectuur te creëeren die Quality of Service (QoS) garandeert bij de distributie van multimedia tussen gebruikers met heterogene netwerken [ENT]. De architectuur behandelt alle aspecten die gebruikers tegenkomen bij de creatie, distributie en consumptie van audiovisuele media. Er wordt geprobeerd om de verschillende aspecten niet te categoriseren per taak maar om deze op een uniforme manier te behandelen. Het project is opgebouwd rond het MPEG-21 multimediaraamwerk.

Het project gebruikt onder andere delen uit de MPEG-21 DIA standaard om de Quality of Service voor de gebruikers te garanderen. Delen van de MPEG-21 standaard worden ook gebruikt voor de beveiling van multimediabronnen.

# Hoofdstuk 7

# Conclusie

Multimedia komt in talloze vormen voor en wordt verspreid over een steeds groeiende heterogeniteit van netwerken. De specificaties van de apparaten die de multimedia consumeren verschillen bovendien aanzienlijk. Alsof dit nog niet genoeg is, worden gebruikers steeds veeleisender. Ze verwachten dat ze met elk apparaat over elk netwerk en op elk tijdstip de aangeboden multimedia kunnen consumeren. Gebruikers zijn vaak pas tevreden als multimedia aan hun specifieke noden is aangepast. Dit concept wordt *Universal Multimedia Access (UMA)* genoemd.

Het MPEG-21 standaardisatiecomité tracht de verschillende delen te herkennen die nodig zijn om dit concept te realiseren. In deze thesis is het tweede en het zevende deel van de standaard besproken die respectievelijk *Digital Item Declaration* en *Digital Item Adaptation* noemen. Het tweede deel introduceert een nieuw begrip: een *Digital Item*. Een Digital Item is een container waarin multimedia en metadata op een hiërarchische manier gestructureerd kan worden. Het is de kern van de hele standaard, elk deel dat nieuwe tools standaardiseert werkt met deze container. Het is ook deze container die tussen de verschillende gebruikers uitgewisseld zal worden. Het zevende deel van de MPEG-21 standaard standaardiseert tools die helpen bij de adaptatie van multimedia. De tools zijn XML-beschrijvingen waarbij de syntax en de semantiek van de elementen en attributen is gestandaardiseerd. Dit laat toe om de tools als metadata in een Digital Item op te nemen. In deze thesis zijn voornamelijk de tools die helpen bij het adapteren van multimediabronnen en het garanderen van Quality of Service voor de gebruiker besproken.

Deze twee delen van de standaard laten toe om een MPEG-21 gebaseerde adaptatie-engine te construeren. De adaptatie-engine kan multimedia op een transparante manier aan de noden van de gebruiker adapteren. De adaptatie-engine is bovendien onafhankelijk van het bestandsformaat dat geädapteerd wordt en van de gebruiker waarvoor geädapteerd wordt. De mogelijkheden van de adaptatie-engine zijn wel sterk afhankelijk van de bestandsformaten.

Als een bestandsformaat veel schaleerbaarheid-opties biedt dan zal de adaptatie-engine dit ook bieden. Met schaleerbaarheid-opties bedoelen we manieren om een bitstream te adapteren door datasegmenten weg te laten en/of kleine wijzigingen door te voeren in de bitstream. Dit heeft zowel voordelen als nadelen. Bestandsformaten die tegenwoordig populair zijn (bijvoorbeeld JPEG, MPEG-4, MP3, ... etc.) bieden weinig schaleerbaarheid-opties. De adaptatie-engine kan multimedia in deze formaten niet of slechts in een beperkte mate adapteren. Er zijn tegenwoordig ook nieuwe bestandsformaten ontwikkeld of in ontwikkeling (bijvoorbeeld JPEG2000, Scalable Video Coding, ... etc.) die betere schaleerbaarheid-opties bieden. De mogelijkheden voor de adaptatie-engine met deze bestandsformaten zijn veel groter zonder dat we de adaptatie-engine zelf hiervoor moeten aanpassen. De adaptatie-engine is ook onafhankelijk van de gebruiker waarvoor geädapteerd wordt. Als invoer voor de adaptatie-engine geven we een UED-beschrijving die het systeem, het netwerk en de gebruiker zelf beschrijft. Op basis hiervan wordt gekozen hoe de multimediabronnen geädapteerd worden.

De tools die in de MPEG-21 DIA standaard zijn ontwikkeld bieden de mogelijkheid om adaptatie-engines te construeren die zowel formaatonafhankelijk zijn als onafhankelijk van de gebruiker zijn. De adaptatie kan bovendien op een volledig transparante manier verlopen. MPEG-21 standaardiseert de adaptatie-engine zelf niet, dit zal zeker voor concurrentie bij de ontwikkeling van adaptatie-engines in de toekomst zorgen. De ontwikkeling van adaptatie-engines zal hierdoor ook actief bijdragen tot de verdere ontwikkeling of uitbreiding van de standaard.

# Bibliografie


[ADA] Adactus. http://www.adactus.no/.

[BDD05] Ian S. Burnett, Stephen J. Davis, and Gerrard M. Drury. Mpeg-21 digital item declaration and identification-principles and compression. *IEEE Transactions on Multimedia*, 7(3):400–407, 2005.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifier (uri): Generic syntax. RFC 3986 (Standard), January 2005.

[BPdWK06] Ian S. Burnett, Fernando Pereira, Rik Van de Walle, and Rob Koenen. *The MPEG-21 Book*. John Wiley & Sons, 2006.

[CBN+07] Petr Cimprich, Oliver Becker, Christian Nentwich, Honza Jirousek, Manos Batsis, Paul Brown, and Michael Kay. Streaming transformations for xml (stx) version 1.0, April 2007. http://stx.sourceforge.net/documents/spec-stx-20070427.html.

[Con04] World Wide Web Consortium. Xml schema specification, 2004. http://www.w3.org/XML/Schema.

[Con06] World Wide Web Consortium. Xml inclusions, 2006. http://www.w3.org/TR/xinclude/.

[DAN] Dynamic and distributed adaptation of scalable multimedia content in a context-aware environment. http://danae.rd.francetelecom.com/.

[DTHH05] Sylvain Devillers, Christian Timmerer, Jörg Heuer, and Hermann Hellwagner. Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, 7(3):463–470, 2005.

[ENI] Enikos. http://www.adactus.no/.

[ENT] Enthrone. http://www.ist-enthrone.org.

[ISO91] ISO/IEC. ISO/IEC 11172: Coding of Moving Pictures and Associated Audio at up to About 1.5 Mbit/s. Technical report, ISO, 1991.

[ISO94a] ISO/IEC. ISO/IEC 13818: Generic Coding of Moving Pictures and Associated Audio. Technical report, ISO, 1994.

[ISO94b] ISO/IEC. ISO/IEC 14496: Coding of Audio-Visual Objects. Technical report, ISO, 1994.

[ISO00] ISO. Information technology – jpeg 2000 image coding system – part 1: Core coding system. Technical report, ISO/IEC, 2000.

[ISO02] ISO/IEC. ISO/IEC 15938: Multimedia Content Description Interface. Technical report, ISO, 2002.

[ISO03] ISO/IEC. Multimedia framework (mpeg-21) – part 7: Vision, digital item adaptation (working draft), 2003.

[ISO04] ISO/IEC. Multimedia framework (mpeg-21) – part 1: Vision, technologies and strategy. Technical report, ISO, 2004.

[ISO05] ISO/IEC. Multimedia framework (mpeg-21) – part 2: Vision, digital item declaration. Technical report, ISO, 2005.

[ISO08] ISO/IEC. Multimedia framework (mpeg-21) – part 8: Reference software, 2008.

[Kay07] Michael Kay. XSL transformations (XSLT) version 2.0. W3C recommendation, W3C, January 2007. http://www.w3.org/TR/2007/REC-xslt20-20070123/.

[KWCK05] Jae-Gon Kim, Yong Wang, Shih-Fu Chang, and Hyung-Myung Kim. An optimal framework of video adaptation and its application to rate adaptation transcoding. *ETRI Journal*, 27(4):341–354, August 2005.

[NWS+00] Gavin Nicol, Lauren Wood, Robert Sutor, Vidur Apparao, Scott Isaacs, Arnaud Le Hors, Chris Wilson, Mike Champion, Jonathan Robie, Steve Byrne, and Ian Jacobs. Document object model (DOM) level 1 specification (second edition). W3C working draft, W3C, September 2000. http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/.

[RHC+06] Michael Ransburg, Hermann Hellwagner, Renaud Cazoulat, Benoit Pellan, Cyril Concolato, Saar De Zutter, Chris Poppe, Rik Van de Walle, and Andreas Hutter. Invited paper: Dynamic and distrubuted adaptation of scalable multimedia content in a context-aware environment. In *Proceedings of the WiCon'06 conference*, page 5, Boston, 8 2006. ACM.

[Tim03] Christian Timmerer. Resource adaptation using xml within the mpeg-21 multimedia framework. Master's thesis, University Klagenfurt, Oostenrijk, 2003.

[Vet04] Anthony Vetro. Mpeg-21 digital item adaptation: Enabling universal multimedia access. *IEEE MultiMedia*, 11(1):84–87, 2004.

[VT05] Anthony Vetro and Christian Timmerer. Digital item adaptation: overview of standardization and research activities. *IEEE Transactions on Multimedia*, 7(3):418–426, 2005.

[XAL] Xalan c++. http://xml.apache.org/xalan-c/.

[XER] Xerces-c++. http://xerces.apache.org/xerces-c/.

[XMLa] Simple api for xml (sax). http://www.saxproject.org/.

[XMLb] Xmlspy. http://www.altova.com/products/xmlspy/xml_editor.html.

[XQI] Xqilla. http://xqilla.sourceforge.net/HomePage.