Software Archivering met Emulatie

Michiel van Dam - 1224239
Jeff van Egmond - 1308041

4 augustus 2010

Executive Summary

Veel culturele en onderzoeksdata wordt tegenwoordig gearchiveerd. Bij zo'n archief is het van belang dat het op lange termijn nog mee gaat, dus dat technologische veranderingen het archief niet ontoegankelijk maken. Voor software en andere dynamische data levert dit speciale problemen op, aangezien het meer afhankelijkheden heeft dan normale data.

Voor archivering van data zijn er op het moment twee bruikbare opties: Migratie en Emulatie. Bij migratie wordt het gearchiveerde bestand omgezet naar een nieuw bestandsformaat, wanneer het oude bestandsformaat ontoegankelijk raakt. Dit heeft als risico's dat er beschadigingen optreden of dat een deel van de layout verloren gaat. De transformatie naar een nieuw bestandsformaat moet ook vaker herhaald worden zodra het 'nieuwe' formaat verouderd. Om de frequentie van zulke transformaties drastisch te verlagen is er binnen DANS een lopend project MIXED, wat zulke data transformeert naar een formaat in XML, wat daardoor niet verouderd en gebruikt zal kunnen blijven worden. Migratie is echter niet geschikt voor software, aangezien de transformatie naar een nieuwer formaat vaak onmogelijk is.

Emulatie is de andere archiveringsstrategie op het moment. In tegenstelling tot Migratie wordt de look-and-feel behouden in de oorspronkelijke omgeving. De twee problemen die hierbij ontstaan is het herkennen welke emulatiemogelijkheid je nodig hebt voor archivering en hoe je de emulatieomgeving zelf onderhoudbaar ontwerpt, zodat de emulator niet door veroudering zal wegvallen. Op dit moment lijkt het laatste nog geen probleem te zijn, aangezien voor elk oud systeem er een emulator op internet te vinden is die op huidige systemen werkt. Het gevaar is echter dat dit over enkele jaren niet meer zo is.

Voor de oplossingen is gekeken naar op Europees niveau lopende projecten en samenwerkingsverbanden. Ten eerste het PLANETS project, sinds juni 2010 de Open Planets Foundation, waarin een framework ontwikkeld is en wordt voor het aanbieden en testen van emulatietools. Hierbinnen loopt ook het KEEP-project dat zich richt op het onderhoudbaar houden van emulatoren of een emulatieframework door ze makkelijk over te kunnen zetten naar andere systemen. Hierbinnen wordt gebruik gemaakt van het Universal Virtual Computer (UVC) principe - een virtuele machine die op alle systemen kan draaien - om de emulatoren ook toegankelijk te houden op lange termijn. Ook wordt voor de bestandsherkenning en voor het vaststellen van de eisen voor archivering gebruik gemaakt van DROID en PRONOM, twee andere onderliggende projecten.

Een oplossing voor het eerste probleem van emulatie wordt benaderd door de zogeheten View Paths, en een Preservation Manager in het KEEP project. Deze onderzoeken de mogelijkheden en vereisten voor het emuleren van een bepaald stuk software of data en geven hierover een conclusie over of iets te emuleren is, en wat er moet gebeuren voordat het geëmuleerd kan worden, mocht het niet zo

zijn. Een oplossing voor het tweede probleem van emulatie kan gebeuren door het UVC principe toe te passen, of gebruik te maken van emulatieframeworks die dat al doen, zoals PLANETS en KEEP als onderdeel daarvan. Hiernaast zijn licenties een mogelijk probleem wat niet verder uitgewerkt is. Er is op dit moment nog niet afdoende onderzoek verricht naar de licentieproblemen bij het emuleren van systemen ten behoeve van archivering (en specifiek toegankelijke gearchiveerde software en/of data).

Een ander belangrijk verschil tussen Emulatie en Migratie zijn de kosten die erbij komen kijken. De kosten voor Emulatie zijn voornamelijk eenmalig, bij het maken van nieuwe emulatoren, terwijl de kosten van migratie periodiek zijn. Dit zorgt ervoor dat ook voor grote hoeveelheden niet-dynamische objecten het op lange termijn goedkoper is om met emulatie te archiveren.

Door het gedane onderzoek zijn een aantal conclusies bereikt. Ten eerste is Emulatie geen keus, maar een must wanneer dynamische objecten gearchiveerd moeten worden. Emulatie is hiernaast een generieke oplossing die werkt voor zowel dynamische als statische objecten, maar voor statische objecten kan migratie handiger zijn als het toegangkelijk moet blijven voor een grote groep mensen. Ten tweede zijn de huidige problemen met emulatie in drie punten samen te vatten: geen off-the-shelf oplossingen, geen duidelijke interface van huidige oplossingen om op verder te werken, en een gebrek aan breed in gebruik zijnde standaarden voor emulatie-oplossingen voor de lange termijn. Aan al deze drie punten wordt binnen de Open Planets Foundation op dit moment al aandacht besteed. Hiernaast is voor emulatie inmiddels al veel onderzoek verricht naar manieren om data opgeslagen te houden, maar nog veel minder naar een manier om de opgeslagen geemuleerde data ook gemakkelijk toegankelijk te maken voor een breed publiek. Hierdoor zijn de manieren voor de zogeheten 'Open Access' minder uitgewerkt dan de manieren van duurzame data-opslag.

Het advies aan DANS is dan ook om vooral niet zelf eigen emulatieoplossingen te ontwikkelen, maar zich aan te sluiten bij de Open Planets Foundation en mee te gaan helpen aan het ontwikkelen van de gestandaardiseerde emulatietools die nodig zijn voor DANS. Een tweede advies is om met het archiveren van software nog een tijdje te wachten tot de tools die nu in ontwikkeling zijn, af zijn. Het derde advies is om in vervolgonderzoek aandacht te besteden aan licenties bij emulatie, en aan 'Open Access', omdat de externe toegang tot de beheerde bestanden van DANS op dit moment een belangrijke eigenschap van DANS lijkt te zijn.

Prototype Aan de hand van het onderzoek is een prototype gemaakt van een beheerapplicatie die inzichtelijk maakt hoe het beheren van een geemuleerd archief kan geschieden. Dit prototype heeft aangetoond dat de meeste afwegingen uit handen van de systeembeheerders genomen kunnen worden waardoor dezen zich kunnen richten op het fysieke beheren van de machines. De status van data-

objecten valt duidelijk in te zien met zulks een beheerapplicatie waardoor tijdig actie kan worden ondernomen. Al met al maakt het prototype de afwegingen die gemaakt worden inzichtelijk en zorgt het voor duidelijkheid dat samenwerken met andere instanties van groot belang is voor het slagen van archivering met emulatie op lange termijn.

Voorwoord

Als onderdeel van ons BSc eindproject aan de Technische Universiteit Delft doen wij onderzoek naar de haalbaarheid van emulatie als lange-termijn archiveringsstrategie voor een instantie als DANS. Naast dit adviesrapport leveren wij ook een prototype beheer-applicatie, waarmee inzicht gegeven kan worden hoe emulatie binnen DANS beheerd zou kunnen worden.

De stagebegeleider vanuit DANS voor dit project is Ir. Rutger Kramer. Vanuit de Technische Universiteit Delft worden we begeleid door Drs. Peter van Nieuwenhuizen. De stage duurde 11 weken.

Planning en verloop

Aan het begin van dit project is een globale planning gemaakt voor de volledige duur van het project. Per week is bepaald wat er ongeveer moest gebeuren. Deze weekplanning is te zien in Appendix C In de linkerkolom staat wat er voor die week gepland stond, in de rechter kolom staan specifiekere gegevens, zoals de exacte datum en tijd van een meeting.

Het eerste deel van het project bestond uit het verrichten van een onderzoek voor DANS, met daarbij het eerste deel van dit rapport. Na een viertal weken verliep het onderzoek doen in het ontwerpen en uiteindelijk ontwikkelen van het prototype.

Besprekingen

Twee-wekelijks waren er besprekingen met Drs. Peter van Nieuwenhuizen en stagebegeleider Ir. Rutger Kramer. Bij deze besprekingen werd de voortgang besproken en eventueel een kleine demo gegeven. Naast deze vooraf geplande besprekingen zijn er regelmatig besprekingen geweest met Ir. Rutger Kramer, bijvoorbeeld om use cases voor het prototype te bedenken.

Daarnaast hebben er meetings plaatsgevonden met Jeffrey van der Hoeven MSc van de Koninklijke Bibliotheek en Maurice van den Dobbelsteen MSc van het Nationaal Archief. Deze meetings waren met name inhoudelijk en bedoeld om meer informatie te winnen voor het onderzoek. Daarbij hebben we ook gesprekken gehad met Dr. Dirk Roorda (DANS), Dr. René van Horik (DANS) en Dr. Henk Koning (DANS). Deze mensen willen we dan ook bedanken voor hun medewerking.

Michiel van Dam,
Jeff van Egmond

Inhoudsopgave

Executive Summary **ii**

Voorwoord **v**

I Onderzoek **8**

1 Inleiding **8**

2 Migratie **10**
2.1 Bestaande oplossingen 10
2.1.1 MIXED 10
2.2 Toekomstbeeld van MIXED 11

3 Emulatie **12**
3.1 UVC 14
3.2 OVF 15
3.3 Oplossingen bij anderen 15
3.3.1 Koninklijke Bibliotheek 15
3.3.2 Nationaal Archief 16
3.4 Ontwikkelingen in de emulatie 16
3.4.1 PLANETS 16
3.4.2 KEEP 17
3.4.3 GRATE 18
3.5 Licenties 18

4 Kosten van archivering **19**

5 Eigen bevindingen **20**
5.1 Emulatie op basis van OVF 20

6 Conclusies **21**
6.1 Waarvoor is emulatie geschikt? 21
6.2 Wat zijn op dit moment de problemen? 21
6.3 Wat zijn de beschikbare mogelijkheden? 22

7 Aanbevelingen voor DANS **22**
7.1 Samenwerken 23
7.2 Nog even wachten 23
7.2.1 OVF als uitzondering 23
7.3 Open Access 23
7.4 Licenties 24

II Implementatie **24**

8 Prototype **24**
8.1 Requirements van het Prototype 24
8.1.1 Prototype I/O 25
8.2 Ontwerp 25
8.2.1 Randvoorwaarden 25
8.2.2 De workflow in het prototype 26
8.2.3 Interne werking 26
8.3 Ontwikkelmethode 30
8.3.1 Testmethode 30
8.3.2 Coverage testing met Emma 30
8.4 Design Decisions 31

9 Bevindingen door prototype **32**

Referenties **34**

III Bijlagen **36**

A Prototype Class Diagrams **37**

B Emma Coverage Report **47**

C Planning **54**

Deel I
Onderzoek

1 Inleiding

In deze sectie wordt een korte toelichting gegeven op DANS en het probleemdomein van softwarearchivering.

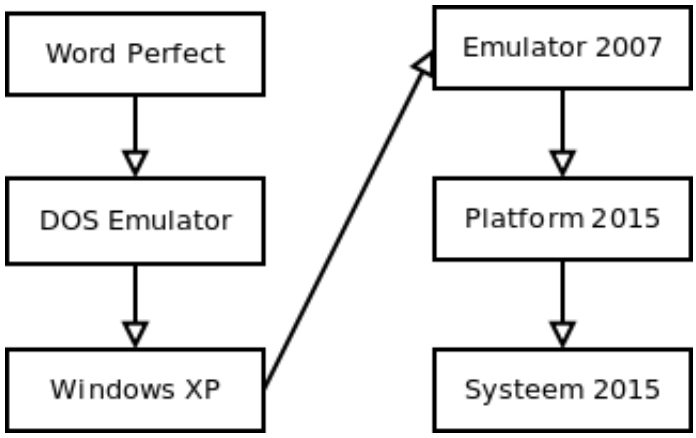
DANS

Data Archiving and Networked Services (DANS) zorgt voor de opslag en blijvende toegankelijkheid van onderzoeksgegevens in de alfa- en gammawetenschappen. Daartoe ontwikkelt het instituut zelf duurzame archiveringsdiensten, bevordert het dat anderen dat doen, en werkt samen met andere databeheerders. Door zijn missie is DANS van nature het Open Access principe toegedaan, maar het heeft ook oog voor het feit dat niet alle data altijd onbeperkt vrij beschikbaar kunnen zijn. Toch is het van belang dat onderzoekgegevens die (nog) niet of beperkt beschikbaar zijn wel duurzaam worden gearchiveerd. Daarom hanteert DANS het principe *Open als het kan, beschermd als het moet.*[1]

Probleemdomein

Bij archivering is het voornamelijk van belang dat het archief duurzaam is. Dat wil zeggen dat zelfs met toekomstige technologische wijzigingen de opgeslagen data nog steeds te raadplegen valt. Voor de meeste data is dit geen probleem, aangezien die als enige afhankelijkheid een specifieke character-encoding heeft, maar voor software kan dit een groter probleem zijn, als deze afhankelijk is van specifieke libraries in een specifieke windows-versie, of van andere platformen die over lange tijd niet meer gebruikt worden. Door de verschillende afhankelijkheden van software en het huidige archiefsysteem van DANS, kan DANS op dit moment nog geen software duurzaam archiveren.

Steeds vaker komt het voor dat niet alleen onderzoeksdata aangeboden wordt om te archiveren, maar ook software. Hierbij gaat het bijvoorbeeld om software, ontwikkeld tijdens een onderzoek, die iets specifieks met deze onderzoeksdata doet. Naast het archiveren van onderzoekssoftware komt het ook voor, of zal in de toekomst voor gaan komen, dat voor bepaalde gearchiveerde data geen software bestaat die de data op huidige systemen toegankelijk maakt. Hierbij kan bijvoorbeeld gedacht worden aan een gesloten formaat waarvoor de gesloten software uit gebruik raakt en na jaren niet meer door de hardware van huidige systemen ondersteund wordt.



Figuur 1: Gestapelde emulatie

DANS wil ervoor zorgen dat software voor deze doeleinden ook duurzaam gearchiveerd kan worden. Hiervoor kan dan gedacht worden aan emulatie als mogelijke archiveringsstrategie. Emulatie in deze zin houdt in dat een platform waarop de bewuste data of de specifieke software in te zien en te gebruiken valt, nagebootst wordt door een ander stuk software.

Een emulator is zelf natuurlijk ook een stuk software. Hierdoor moet er bij de emulatie-oplossing goed gekeken worden naar de emulatoren en de manier om deze ook langdurig te kunnen bewaren of gebruiken, zodat het niet of slechts beperkt nodig is om omgevingen te emuleren voor emulatoren, zoals afgebeeld in figuur 1.

Voor het bepalen of emulatie haalbaar is voor DANS, zijn een aantal vragen van belang:

- Voor welke data is het echt nodig;
- Welke oplossingen worden er in de nabije toekomst afgemaakt die voor deze data werken;
- Welke juridische en financiële afwegingen zijn er voor DANS (Welke juridische en financiële haken en ogen zitten er aan de mogelijke vormen van emulatie).

In dit verslag zullen we beginnen met een overzicht van de huidige en geplande technieken op het gebied van migratie, de huidige en geplande technieken op het gebied van emulatie, en hoe deze technieken voor het archiveren van software kunnen helpen. Hierna bespreken we kort de kostenafwegingen die er bij emulatie ten opzichte van migratie bestaan, alvorens we conclusies trekken uit het behandelde en een koers voor DANS aanraden.

2 Migratie

Op dit moment is migratie de geprefereerde archiveringsstrategie bij DANS. In deze sectie zullen eerst de mogelijkheden en afwegingen bij migratie besproken worden, en daarna volgt een overzicht van de huidige en geplande ontwikkelingen op dit gebied. Hierbij zal gekeken worden naar het DANS- project MIXED en de preservation planning tool PLATO. Daarnaast zal kort gekeken worden naar het toekomstbeeld dat migratie biedt.

Bij het migreren van data worden bestandsformaten geconverteerd naar nieuwe bestandsformaten, op het moment dat de oorspronkelijke formaten verouderd dreigen te raken. Als bijvoorbeeld het Word 95 formaat binnenkort niet meer toegankelijk gaat zijn, zullen de documenten naar bijvoorbeeld het Word 2007 formaat overgezet moeten worden. Op deze manier blijven de documenten bewaard om ook na het verouderen van Word 95 nog toegankelijk te zijn. Zodra Word 2007 op zijn beurt dan verouderd raakt, zullen de documenten opnieuw geconverteerd moeten worden.

Het voordeel hiervan is dat documenten altijd beschikbaar zijn in een huidig toegankelijk en breed geaccepteerd formaat, zoals PDF, waardoor huidige software het zonder veel moeite kan weergeven. Een belangrijk nadeel is echter dat bij het converteren van documenten layout of in het slechtste geval data verloren kan gaan. Als het bewaren van de 'look and feel' belangrijk is, of wanneer het om dynamische objecten gaat, is migratie waarschijnlijk niet de beste oplossing[2]. Hierdoor moet bij het kiezen van een archiveringsmethode overwogen worden welke functionaliteit behouden dient te blijven. Moeten gegevens bijvoorbeeld alleen nog in te zien zijn, moeten data-objecten nog te bewerken zijn of is read-only toegang genoeg, of is de 'look and feel' belangrijk?

2.1 Bestaande oplossingen

Momenteel zijn er al een aantal projecten op het gebied van migratie gaande. Binnen DANS is daarvan MIXED het beste voorbeeld.

2.1.1 MIXED - Migration to Intermediate XML for Electronic Data

Een belangrijk lopend migratieproject is Migration to Intermediate XML for Electronic Data (MIXED)[3], van DANS. Binnen MIXED wordt data gemigreerd naar SDFP (Sustainable Data Format for Preservation) of M-XML (MIXED-XML). Wanneer deze data dan weer gebruikt moet worden, is het mogelijk met een conversieprogramma de data te converteren naar een formaat dat door hedendaagse software ondersteund wordt. Het ontwikkelen van die conversieprogramma's valt ook binnen MIXED. Door het gebruik van een intermediair dataformaat is het niet nodig om bij elke verandering van 'huidige' software de data te migreren, er hoeft 'slechts' een nieuwe convertor geschreven te worden. Dit laatste is aanzienlijk minder tijdrovend dan het met grote regelmaat migre-

ren van de data zelf. Bovendien is de kans dat er verlies van data optreedt veel kleiner.

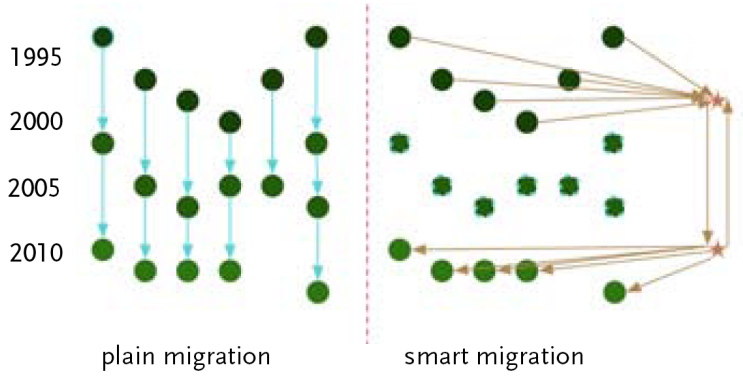
Het verschil tussen formaat en data Binnen gegevensarchivering is het makkelijk om te denken over dataformaten als de bestandstypen die ze hebben. Denk hierbij aan een Microsoft Excelsheet .xls, .doc, .tif, en dergelijken. Hoewel dit belangrijk kan zijn voor het migreren naar een preferred format, is het ook belangrijk om het bestandstype los te zien van de datasoort. Bij datasoort kan gedacht worden aan tabeldata, tekstdocumenten, afbeeldingen, etc. Dit is belangrijk omdat er voor elk van deze datasoorten algemeen gedeelde aannames over de inhoud zijn. Bij tekstdocumenten is breed bekend dat je tekst in een bepaald lettertype op een bepaalde puntsgrootte hebt, dat sommige tekst vet, cursief of onderstreept is, dat er een koptekst en voettekst kan zijn en dat je een inhoudsopgave in een document hebt. Dankzij deze aannames kan een tekst gearchiveerd worden zonder exact de vorm van het document te behouden, maar met behoud van gedetailleerde inhoud. Vaak is dit voor archivering namelijk al genoeg. Het project MIXED (en het SDFP formaat) van DANS is op dit moment geschikt voor datasoorten waarover al een duidelijk paradigma bestaat en waarvan de formaten al doorontwikkeld zijn. Zodra dit paradigma bestaat kan er een migratietool geschreven worden die duurzame XML oplevert.

Hierdoor is MIXED echter niet geschikt voor toepassing op Software of Video of documenten waarvan de vorm juist belangrijk is. Ook voor andere dataformaten is een dergelijke (XML-based) migratie echter niet altijd geschikt. Door het converteren van Bitmap-afbeeldingen naar XML zou het benodigde aantal bytes om het object te omschrijven explosief groeien. In zulke gevallen lijkt een vorm van 'runtime-conversie' of emulatie beter geschikt dan migratie.

2.2 Toekomstbeeld van MIXED

MIXED wordt nog steeds doorontwikkeld. De definities van het gekozen intermediaire formaat zijn wellicht niet dekkend voor toekomstige digitale (data)objecten. In dat geval zal op een gegeven moment de definitie van M-XML (of het nieuwere SDFP) aangepast moeten worden. Wanneer dit gebeurt worden niet alle gearchiveerde objecten direct gemigreerd naar het nieuwe formaat, zoals te zien in figuur 2. In plaats daarvan wordt een conversieprogramma geschreven van het oude M-XML naar het nieuwe M-XML formaat en vice versa. Doordat het vernieuwen van definities in veel gevallen slechts inhoudt dat er nieuwe definities bij komen en de oude blijven bestaan, verandert een M-XML bestand bij een conversie van het oude naar het nieuwe formaat in veel gevallen niet. Een conversieprogramma van het nieuwe formaat naar het oude formaat is dan ook triviaal: alle 'nieuwe' definities worden gefilterd.

Dat het converteren van oud naar nieuw noodzakelijk is is duidelijk, maar waarom wordt er ook een conversieprogramma van nieuw naar oud gemaakt? Wanneer besloten wordt het intermediaire formaat aan te passen, heeft dat



Figuur 2: Migratie volgens MIXED[3]

meestal te maken met nieuwe functionaliteiten van nieuwe software die met de data om kan gaan. Het is echter niet zo dat onderzoekers die die data willen gebruiken per se de nieuwste software bezitten. Voor het oude formaat bestond een conversieprogramma naar die oude software. Door een conversieprogramma van nieuw naar oud formaat te ontwikkelen, wordt daarmee de data voor oude software, op zo correct mogelijke wijze, toegankelijk gehouden. Voor het nieuwe formaat worden verder alleen conversieprogramma's ontwikkeld naar huidige software.

Na verloop van tijd is het dan zo dat voor een bepaald oud M-XML formaat geen conversieprogramma's meer bestaan, die het naar een formaat converteert dat door huidige soft- en of hardware wordt ondersteund. In zo'n geval kan besloten worden dat oude M-XML formaat te laten vervallen. Dit betekent dan wel dat alle data, die in dat oude formaat is gearchiveerd, in één keer geconverteerd moet worden naar een nieuwer formaat dat op dat moment nog een lange levensduur heeft[3].

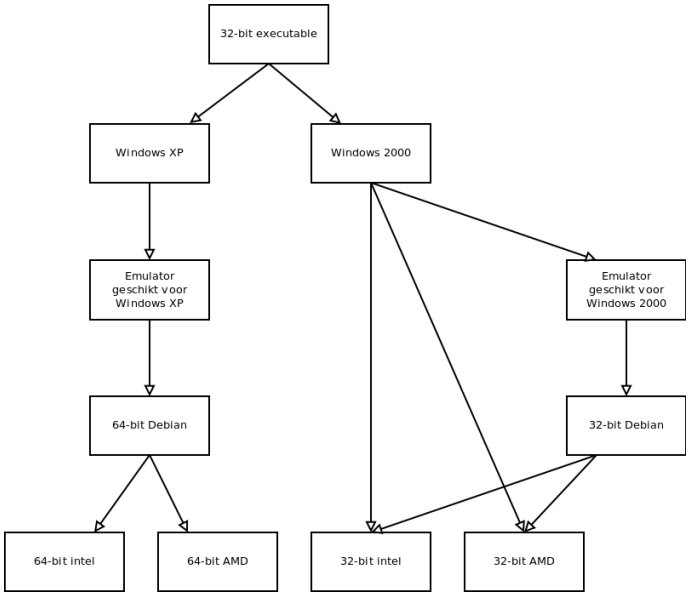
3 Emulatie

Een alternatief voor migratie is emulatie. Emulatie behoudt, in tegenstelling tot migratie, het oorspronkelijke document en geeft de mogelijkheid 'oude' software te gebruiken om de bestanden te bekijken. Emulatiesoftware geeft hiermee authentieke weergave, door de oorspronkelijke software op het oorspronkelijke platform te gebruiken in toekomstige omgevingen[4].

Het voordeel is dat de look-and-feel van de opgeslagen data behouden blijft. Een belangrijk nadeel is echter dat het maken en onderhouden van emulatiesoftware veel tijd kost. In de toekomst zullen meerdere emulatieomgevingen on-

derhouden moeten worden, en er kan niet zonder meer gezegd worden dat deze emulatieomgevingen op alle toekomstige systemen nog steeds zullen werken[5]. Dit tijdrovende onderhoud kan echter ingeperkt worden door van een virtuele laag in de emulatieomgevingen gebruik te maken[6]. Emulatiesoftware wordt dan geschikt gemaakt om op die virtuele machine te werken, waarbij de interface van de virtuele machine hetzelfde blijft, ongeacht het platform waar die virtuele machine op staat. Eens in de zoveel tijd zal dan wel de virtuele machine aangepast moeten worden aan nieuwe systemen, maar op deze manier blijft de emulatiesoftware functioneel zonder verder onderhoud.

View paths Voor emulatie is het van belang kennis te hebben van het principe View paths[7]. Een view path omschrijft een manier om tot weergave van een bestand of programma te komen. Neem bijvoorbeeld een 32-bits executable, zoals in figuur 3. Volgens deze illustratieve schets zijn er zes paden om een 32-bits executable uit te voeren.



Figuur 3: Een voorbeeld van View paths

Om een idee te hebben van (het gebrek aan) ondersteuning voor een bepaald bestandstype of programma, kan onder andere gekeken worden naar het aantal beschikbare View paths. Daarnaast kan het bijvoorbeeld zo zijn dat één stuk software (zoals Windows 2000) in een groot deel van de viewpaths voor een bepaald bestandstype voorkomt. Zo'n stuk software is dan van wezenlijk belang voor de ondersteuning van dat bestandstype.

De emulator emuleren Wanneer gekeken wordt naar emulatie als strategie voor het archiveren van software, is het belangrijk rekening te houden met het feit dat een emulator zelf ook een stuk software is. Omdat een emulator zelf een stuk software is, heeft een emulator zelf ook afhankelijkheden, een bepaalde emulator draait bijvoorbeeld alleen op 32-bits Windows XP systemen.
Het is mogelijk om emulatoren te emuleren en zo een stapel ('stack') van emulatie te vormen, zoals in figuur 1.[8] Op deze manier kunnen oude data-objecten in theorie altijd beschikbaar gehouden worden. Emuleren geeft echter een last op de performance van een systeem, waardoor het stapelen van emulator op emulator een groter performance probleem kan zijn. Daarnaast ontstaat hier het probleem van veel afhankelijkheden. Over 50 tot 100 jaar zijn een groot aantal werkende emulatoren nodig om hedendaagse software weer te geven.
Doordat het stapelen van emulatoren een groot performance probleem kan of zelfs zal vormen, is het belangrijk een oplossing te zoeken die voorkomt dat emulatoren geëmuleerd moeten worden om data-objecten beschikbaar te houden.

3.1 UVC - Universal Virtual Computer

In 2001 poneerde Lorie[5] het idee van een Universal Virtual Computer (UVC). Een UVC is functioneel gelijk aan een computer, maar dan virtueel en gedefinieerd op een wijze die 'voor altijd' mee kan gaan. Lorie stelt voor een architectuur te maken met een zo klein mogelijke instructieset. In theorie kan de UVC een oneindig aantal registers hebben, waarbinnen adressering bit-oriented is: een 8-bit per byte computer kan dan net zo makkelijk geëmuleerd worden als een 9-bit per byte computer. Voor de UVC is snelheid niet van belang, omdat logischerwijs computers in de verre toekomst (stel: 100 jaar van nu) extreem veel sneller zijn dan huidige computers. Om een stuk data, opgeslagen als bit-stream, te kunnen weergeven zijn dan een aantal programma's, ontwikkeld voor de UVC, nodig:

- een restore program om de bit-stream uit te lezen en in een virtueel geheugen op te slaan
- een UVC interpreter

Logical Data View Een restore program voor de UVC slaat de bit-stream op in een virtueel geheugen. Daarvoor is een formaat nodig: de Logical Data View (LDV)[7]. Een LDV is een schematisch formaat, bijvoorbeeld een XML formaat. Dit betekent voor bijvoorbeeld bitmap afbeeldingen dat een LDV hiervoor veel meer bits nodig heeft dan het originele formaat. Dat is een van de redenen om de originele bit-stream op te slaan. Een UVC interpreter voor een bepaald formaat hoeft slechts een LDV te kunnen interpreteren. Wanneer men als LDV voor een bitmap afbeelding een XML formaat heeft, zijn er meerdere interpreters te bedenken, voor verschillende weergaves:

- Een afbeelding weergave

- Een XML-weergave

Door het simpele ontwerp, met een beperkte instructieset, van de UVC, is het in theorie voor een software-ontwikkelaar weinig werk om op een nieuw type toekomstige computer een UVC-implementatie te maken.

Alle bestaande UVC implementaties zijn echter nog zeer beperkt, en voor het gemak gebouwd op de Java Virtual Machine (JVM) die een goede UVM benadering is. Projecten zoals Dioscuri[9] zijn bedoeld als een proof-of-concept, maar geenszins bruikbaar voor brede inzet. Hier moet KEEP (zie 3.4.2) echter verandering in gaan brengen.[10]

Op dit moment zijn er al veel emulatoren die een platform kunnen virtualiseren of emuleren. Voorbeelden hiervan zijn SIMH, Bochs, QEMU, VirtualPC, en bekende pakketen zoals VirtualBox en VMWare[8]. Het is in theorie mogelijk om van elk veelgebruikt systeem een virtuele machine te maken, die dan tot in lengte van dagen zijn taken kan doen. Echter, de meeste huidige emulatoren of virtualisatiesoftware vereisen externe drivers of patches voor optimaal functioneren in combinatie met de hardware van het host-platform. Dit gaat de portabiliteit (en daardoor de haalbaarheid van lange-termijn archivering) tegen.

3.2 OVF - Open Virtualization Format

Sinds 2008 is er wel het Open Virtualization Format (OVF)[11] waarmee virtuele machines uitwisselbaar zijn tussen softwarepakketten voor virtualisatie[12]. Dit zorgt ervoor dat op elk systeem met elk OVF-ondersteunend stuk software, een virtuele machine kan draaien. Hierdoor is het in principe ook mogelijk om met een UVC deze virtuele machines uit te voeren. Het idee om van ieder hedendaags platform een virtuele machine te maken waardoor software uitvoerbaar blijft, is hierdoor zeker niet onmogelijk.

3.3 Oplossingen bij anderen

Er zijn natuurlijk al andere mensen in de wereld die met het probleem van duurzame archivering van software te maken hebben. Deze mensen hebben wellicht eigen systemen waarmee ze software nu archiveren of gearchiveerd hebben. Een klein overzichtje van de partijen die wij gesproken hebben volgt hieronder.

3.3.1 Koninklijke Bibliotheek

Op dit moment wordt bij de Koninklijke Bibliotheek (KB) al wat gedaan aan het archiveren van software, aangezien de KB regelmatig educatiesoftware of proof-of-concept software bij proefschriften krijgt. Een paar jaar terug heeft een project gelopen waarbij ingekomen software geinstalleerd werd op een 'reference workstation', en er vervolgens een virtuele machine gemaakt werd voor deze workstation. Deze virtuele machines werden dan vervolgens opgeslagen in

het e-Depot van de KB. Omdat deze machines vaak meer dan een Gigabyte besloegen en uitgevoerd moesten worden op specifieke PC's, is dit project ter ziele gegaan en tegenwoordig wordt ingekomen software bij de KB op digitale dragers opgeslagen zonder duurzame inzagemogelijkheid, voor het moment waarop zo'n inzagemogelijkheid wel zal ontstaan.

3.3.2 Nationaal Archief

Het Nationaal Archief beheert op dit moment al enkele documenten waarbij de 'look-and-feel' belangrijk is. Dit gebeurt op dit moment echter nog niet binnen een emulatieframework maar door het opslaan in open beeldformaten zoals PNG. Een paar ontwikkelingen waar het Nationaal Archief bij betrokken is zullen in de aankomende secties besproken worden.

3.4 Ontwikkelingen in de emulatie

Een aantal projecten die momenteel lopen, bijna aflopen of recent afgelopen zijn, worden bekeken om aan te geven wat emulatie in de toekomst kan betekenen. Er zal onder andere gekeken worden naar de projecten PLANETS, KEEP en GRATE.

3.4.1 PLANETS - Preservation and Long-term Access through Networked Services

Op het moment van schrijven is het Nationaal Archief verwikkeld in een Europees project 'Preservation and Long-term Access through Networked Services' (PLANETS)[13]. PLANETS is een project met als doel practische services en tools te produceren om het op lange termijn beschikbaar houden van data te ondersteunen. PLANETS heeft voor emulatie een aantal goede stappen gezet. Het voornaamste zijn een standaardisering en een framework waarbinnen eigen emulatoren opgehangen kunnen worden. Elke emulator moet een aantal standaarden ondersteunen zoals Create View, en dan kan het aansluiten op PLANETS. Dioscuri was hier al een eerste voorbeeld van. Dioscuri is geheel modulair opgebouwd, en het is ook de insteek van de rest van het PLANETS project geweest om het op dezelfde manier modulair te houden. Waar er binnen Dioscuri al emulatoren voor DOS geschreven waren, is KEEP nu ook bezig om voor sommige Windows platformen een emulator te ontwikkelen. Binnen de Open Planets Foundation (OPF)[14] worden deze module-standaarden verder gebruikt en wordt er regelmatig overlegd met onder andere Microsoft Research en IBM. Binnen PLANETS wordt er ook een wrapper om bestaande emulatoren heen gebouwd zodat ze in het OPF framework passen.

PLANETS heeft een Testbed beschikbaar voor het benchmarken van tools. Hierbinnen is er de planningstool PLATO, die aan de hand van metadata voor en na conversies adviseert over migratie of emulatiestappen. Met een

testbed-account kun je hier gebruik van maken. De bestanden worden eerst door DROID[15] beoordeeld en er wordt een suggestie over het formaat gedaan. Vervolgens gaat JHOVE[16] kritisch bekijken of het bestand ook echt van dat formaat is. Als een bestand door deze test heen komt heet het dan JHOVE-gevalideerd. Een ander voorbeeld hiervan is PRONOM[17], een pakket wat van bestanden kan zeggen welk bestandformaat het heeft en welke programma's geschikt zijn om deze formaten te openen. Hiervoor maakt het ook gebruik van DROID voor de bestandsformaten.

PRONOM PRONOM kan bestandstypen herkennen en software benoemen die met deze bestandstypen om kan gaan. Voor het bijhouden van bestandstypen worden PRONOM Unique Identifiers (PUID) gebruikt. Ieder bestandsformaat krijgt een eigen PUID aan de hand waarvan ze te herkennen zijn. Op dit moment zijn de meest gebruikte 130 formaten al voorzien van een PUID in de PRONOM database, en er blijven er regelmatig meer bij komen. Als onderdeel van PRONOM worden ook enkele behulpzame tools ontwikkeld. Hiervan is de eerste al af: DROID.

Naast de herkenning van bestandsformaten kan PRONOM ook ingezet worden voor opvragen welke software gebruikt kan worden voor het inzien en aanpassen van de gevonden bestandsformaten. Dit maakt het een waardevolle toepassing om automatisch afhankelijkheden bij archivering tegen te komen.

DROID - Digital Record Object Identification Digital Record Object Identification (DROID) is een softwaretool voor het geautomatiseerd herkennen van bestandsformaten. DROID is platformonafhankelijk en geschreven in java. De broncode is vrij verkrijgbaar onder een BSD Licentie. Hierdoor is het relatief gemakkelijk om DROID binnen een ander softwarepakket of organisatie te gebruiken.

3.4.2 KEEP - Keeping Emulation Environments Portable

Keeping Emulation Environments Portable (KEEP) is een europees project onder het overkoepelende PLANETS project[10]. Binnen Nederland is onder andere de Koninklijke Bibliotheek (KB) al betrokken bij de ontwikkeling hiervan. Op dit moment wordt er binnen KEEP hard gewerkt aan een emulation framework wat alles omhelst van het binnenhalen en opslaan van data met metadata tot het flexibel kunnen kiezen van een platform en de software die gezamenlijk geëmuleerd moeten worden.

Met het Transfer Tools Framework kan data van fysieke dragers verwerkt worden en kan het oorspronkelijke object samen met gegenereerde metadata in het digitale archief gezet worden. Met een Preservation Manager wordt vervolgens gekeken welke bijhorende software hiervoor nodig is. Vervolgens kan er beredeneerd worden welke emulatoren hiervoor geschikt kunnen zijn. Bij het opvragen van het data-object kan een persoon via het KEEP Emulation Framework dan

zelf kiezen voor een emulatie-versie, die draait op de KEEP Universal Layer.

Deze Universal Layer is op een vijftal lagen gespecificeerd. Zodra een van de lagen geïmplementeerd wordt, kan KEEP in principe prima erop uitgevoerd worden. De onderste laag heeft maar 2 processor- instructies om te implementeren, wat het toegankelijk maakt tegen een prijs: dit levert geen grote snelheid op. Laag twee heeft al 8 instructies en de vijfde laag is vergelijkbaar met de JVM, qua reikwijdte van de instructies. Waar de onderste laag weinig investering vereist om te implementeren, bieden de hogere lagen een betere performance. Daar staat tegenover dat het implementeren van die lagen op een nieuw systeem meer investering van tijd en moeite vereist.

Voor een optimale werking van KEEP is het van belang een software-archief op te bouwen, met software die binnen de emulatie-omgevingen gebruikt kan worden. Van deze software moeten ook de dependencies qua platform en andere software (zoals libraries) beheerd worden, samen met de bestandstypen die door deze software geopend kunnen worden. De KB bewaart op dit moment nog geen gebruikerssoftware, maar zij kan dit in de toekomst best gaan doen. Dan gaat het voornamelijk om algemeen gebruikte software, niet om specialistische software (zoals onderzoekers die een speciale tool hebben gebruikt op hun dataset). Dit soort specialistische software zou elders ondergebracht kunnen worden (o.a. in samenwerking met DANS). Het KEEP project loopt ten einde in februari 2012.

3.4.3 GRATE - Global Remote Access To Emulation-services

Global Remote Access To Emulation-services (GRATE) is een onderdeel van het PLANETS-project. Door middel van GRATE is het mogelijk om via de webbrowser een viewport naar een emulatieomgeving, die elders gehost is, te verkrijgen[18]. Door middel van GRATE kan wereldwijd toegang gegeven worden tot emulatiesoftware. Momenteel is GRATE echter nog nergens in gebruik.

3.5 Licenties

Voor het emuleren van operating systems of het aanleggen van een software-archief vormen licenties een belangrijk probleem. Zeker wanneer een gearchiveerd stuk software door veel mensen gebruikt kan worden, zal dit juridische problemen op gaan leveren. Ook bij operating systems zullen er de nodige licentiekosten betaald moeten worden voor het (mogelijk massale) gebruik ervan. Echter, voor de te archiveren (onderzoeks)software zal dit minder een probleem vormen, aangezien de eigenaar van de software opdracht geeft tot archivering. Van der Hoeven publiceert in september een paper voor de IPRES-conferentie, waarin hij dieper op dit onderwerp in zal gaan.

4 Kosten van archivering

Om uitspraken te kunnen doen over de haalbaarheid van emulatie als archiveringsstrategie moet er tenminste kort ingegaan worden op de kosten en de afwegingen tussen migratie en emulatie.

Bij migratie komen de voornaamste kosten voort uit het overzetten van bestanden naar een ander formaat, en bij emulatie blijven deze kosten bespaard. Echter bij emulatie is er een grotere initiële investering noodzakelijk, wat het ongeschikt maakt voor opslag op een kortere termijn. De volgende kosten kunnen onderscheiden worden voor data-archivering.[2]

- **Initiële kosten**
  Bij emulatie als strategie moet er een systeem voor emulatoren gemaakt worden, waar emulatoren langdurig op kunnen draaien. Voor migratie hoeven er geen initiële kosten gemaakt te worden.

- **Periodieke kosten**
  Bij emulatie moeten er, waar nodig, nieuwe emulatoren ontwikkeld worden voor systemen die geëmuleerd dienen te worden. Aangezien er nieuwe systemen ontwikkeld blijven worden, kan dit onder periodieke kosten geschaard worden. De frequentie van deze kosten is echter niet hoog, doordat de meeste systemen die op hetzelfde moment in gebruik zijn, compatible zijn. De periodieke kosten voor migratie zijn uiteraard het overzetten naar een nieuwere versie van het bestandsformaat of het overzetten naar een ander bestandsformaat. Dit dient regelmatig te gebeuren zodat altijd breed beschikbare softwarepakketten gebruikt kunnen blijven worden voor het inzien van deze bestanden.
  Voor emulatoren is ook onderhoud nodig, door het jaar heen, in tegenstelling tot losse gemigreerde bestanden.

- **Opvragingskosten en opslagkosten**
  Voor het beschikbaar stellen van de resultaten zal een web-service of benaderbare omgeving moeten draaien, en zal opslagruimte beschikbaar moeten zijn voor het opslaan van de data-objecten. Deze kosten zijn echter vergelijkbaar voor emulatie en migratie, en zullen verder dan ook niet behandeld worden.[2]

Van de eenmalige kosten voor het maken van een emulatorensysteem kan de schatting van Oltmans gebruikt worden van $200.000. De jaarlijkse kosten voor het ontwikkelen van nieuwe emulatoren en het onderhouden ervan zijn volgens Oltmans $30.000. Het grote voordeel van deze aanpak is echter dat deze kosten niet stijgen naarmate het aantal beheerde databestanden toeneemt.

Hiertegenover staan de periodieke kosten van migratie. Voor het migreren van een enkel bestand wordt door Oltmans $0,10 per jaar genoemd. Dit zorgt ervoor dat Migratie erg aantrekkelijk is voor kleine hoeveelheden bestanden of

voor een klein aantal jaren, maar dat voor enorme hoeveelheden bestanden en voor lange termijn Emulatie goedkoper kan zijn. Een simpele rekensom leert dat met bovenstaande bedragen Emulatie goedkoper is bij 50 jaar geplande opslag en 340000 data-objecten, of bij 20 jaar geplande opslag en 400000 data-objecten.

Wanneer het niet gaat over alle mogelijke data-objecten maar specifiek over Software, is dit verhaal al niet meer terzakedoende. Software migreren is een ondoenlijke opgave, zeker in grote hoeveelheden. Bovenstaande kostenargument dient dan ook voornamelijk om een prijskader te schetsen voor emulatie en als argument voor het emuleren van andere data-objecten, als eenmaal gekozen is voor het emuleren ten behoeve van softwarearchivering. Hierbij dient ook nog de opmerking geplaatst te worden dat de initiële kosten van $200.000 niet geheel door DANS gemaakt zullen hoeven worden, gelet op het aankomende KEEP framework, en dat de periodieke kosten voor het ontwikkelen van nieuwe emulatoren voor dit framework waarschijnlijk gedeeld kunnen worden tussen verschillende organisaties en belanghebbenden.

5 Eigen bevindingen

Naast de lopende projecten zijn wij zelf ook met een aantal ideeen gekomen. De belangrijkste was een strategie op basis van het eerder beschreven Open Virtualisation Format (OVF).

5.1 Emulatie op basis van OVF

Het Open Virtualisation Format kan aangegrepen worden om een emulatie-based archiveringsstrategie te ontwikkelen. Dit zou dan volgens een stappenplan moeten gaan:

1. Ontwikkel van alle noemenswaardige huidige systemen een (minimale) generieke virtuele machine en sla deze op in OVF formaat.

2. Ga van te archiveren data-objecten na op welke van deze generieke systemen het draait of met welke software deze toegankelijk is.

3. Installeer of plaats de data op de juiste virtuele machine.

Wanneer nieuwe (besturings)systemen verschijnen, is het dan van belang zo'n systeem even in gebruik te laten komen alvorens deze in OVF-formaat op te slaan. Zo is het mogelijk ervan verzekerd te zijn dat allerlei randzaken zoals codecs ontwikkeld zijn voor dat systeem. Wanneer een nieuw systeem voldoende ondersteund wordt, kan een generieke installatie van dat systeem in OVF opgeslagen worden. Alhoewel grote (besturings)systemen redelijk vaak, tussen de 1 en 2 jaar, met nieuwe versies komen, is het relatief goedkoop om dit te onderhouden. In theorie hoeven oudere systemen die in OVF opgeslagen zijn namelijk niet onderhouden te worden.

Een probleem met deze oplossing is dat het valt of staat met het ondersteunen van het OVF-formaat. Rothenberg schreef al dat het vertrouwen op standaarden die 'eeuwig' zouden zijn niet realistisch is[6]. Dit geldt natuurlijk niet alleen voor standaarden voor dataformaten, maar ook voor standaarden voor virtualisatie, of welke andere standaard dan ook.

6 Conclusies

Op basis van het hiervoor gepresenteerde kunnen er conclusies worden getrokken over de mogelijkheden voor DANS en de haalbaarheid van deze mogelijkheden.

6.1 Waarvoor is emulatie geschikt?

Het blijkt dat Emulatie als archiveringsstrategie niet zonder meer voor alle data geschikt is. Emulatie is vooral nuttig voor dynamische data-objecten als Software, hoewel het ook toe te passen is op statische objecten. Voor statische objecten kan het migratiekosten besparen, maar het brengt licentiekosten en het ontwikkelen van emulatoren met zich mee. Zodra deze emulatoren echter al gemaakt worden voor dynamische data-objecten, wordt het eenvoudiger om de statische objecten hier ook te archiveren. Ook dient er rekening gehouden te worden met het feit dat migratie per data-object per tijdseenheid kosten met zich meebrengt, terwijl emulatie slechts per emulator eenmalige kosten oplevert. Hierdoor kan er geconcludeerd worden dat voor kleine hoeveelheden data emulatie een dure optie is, maar dat voor grote hoeveelheden het voordeliger uit kan pakken dan het migreren. Voor dynamische objecten is het echter geen vrijblijvende keuze, maar de enige mogelijkheid.

MIXED brengt hier slechts lichte verandering in. Dankzij MIXED is migratie van de betrokken data-objecten minder vaak nodig en met minder risico gepaard. Dit zorgt ervoor dat er veel meer data-objecten nodig zijn voordat emulatie hier de goedkopere optie is.

6.2 Wat zijn op dit moment de problemen?

Voor het archiveren met emulatie hebben we een aantal problemen geïdentificeerd. Ten eerste dat van standaarden. Er zijn legio pakketten zoals VMWare, VirtualBOX e.d. beschikbaar waarmee een systeem of operating system geëmuleerd kan worden. De resulterende emulatoren zijn echter veelal niet uitwisselbaar tussen de verschillende softwarepakketten, doordat elk een eigen standaard gebruikt. Ook breder gedragen standaarden kunnen in het geding komen, waardoor het gebruiken van een 'standaard' geen garantie is voor de tijd gedurende welke de opgeslagen gegevens nog met deze standaard in te zien vallen.

Hiernaast ontbreken op dit moment de off-the-shelf oplossingen voor emulatie, wat hoge opstartkosten veroorzaakt op het moment dat emulatie gekozen

wordt als archiveringsstrategie. Ook beschikbare tools zoals PRONOM missen nog een duidelijke manier waarop er geautomatiseerd gebruik van gemaakt kan worden.

Als derde is het moeilijk een verouderd systeem achteraf te gaan emuleren. De hardware waar het verouderde systeem voor geschreven was hoeft niet meer beschikbaar te zijn en de specifieke drivers of ondersteunende software hoeft ook niet meer aangeboden te worden.

Tenslotte zijn de licenties een groot probleem voor het toegankelijk maken van oude data door middel van emulatie. Dit zal echter in mindere mate optreden bij het archiveren van software, omdat de eigenaar van die software toestemming zal geven voor emulatie.

6.3 Wat zijn de beschikbare mogelijkheden?

Er vallen de aankomende tijd een aantal mogelijkheden te identificeren die kunnen leiden tot een goede archiveringsmethode met emulatie. Ten eerste kunnen beschikbare tools zoals PRONOM breder toegankelijk gemaakt worden met een API die gebruikt kan worden door externe projecten. Er zijn, zoals eerder genoemd, veel tools beschikbaar die nuttig zijn, alleen de manier waarop ze te bereiken vallen laat te wensen over. Dit ligt ook in lijn met het standaardiseren van tools. Er worden nu al wrappers voor bestaande emulatoren gemaakt waardoor ze binnen het PLANETS framework passen. Hierdoor worden deze emulatoren op een standaardmanier benaderbaar, wat hierdoor langer mee zal gaan. Het KEEP-project biedt hier dan de definitieve en meestbelovende mogelijkheid om de toegankelijkheid van emulatoren op lange termijn te waarborgen, en hiermee emulatie als archiveringsstrategie te valideren. Hierbij kan dan een remote framework zoals GRATE gebruikt worden voor het inzien van bestanden door klanten.

Binnen een hierboven beschreven systeem kunnen emulatoren blijven werken, maar hier is het aanmaken van de emulatoren dan nog een punt. Als pas na lange termijn oude systemen gepoogd worden met emulatie toegankelijk te maken, zullen sommige drivers of specifieke ondersteunende software niet meer beschikbaar zijn. Hierdoor is het van belang om emulatoren te maken terwijl de huidige systemen nog net ondersteund worden. Dit verlaagt de kosten die benodigd zijn voor het succesvol creëren van een geschikte emulator.

7 Aanbevelingen voor DANS

Aan de hand van de bevindingen die wij hebben gedaan op het gebied van softwarearchivering met emulatie, kunnen we DANS de volgende adviezen geven.

7.1 Samenwerken

Het voornaamste advies luidt om nauwer samen te gaan werken met instanties die al hard bezig zijn om archivering met emulatie op brede schaal mogelijk te maken. Te denken valt aan het Europese PLANETS- of KEEP-project, het zou een goed idee zijn hierbij toenadering te zoeken en proactief mee te helpen bij het ontwikkelen van emulatoren. De emulatoren die binnen het PLANETS framework ontwikkeld worden zullen ook door een brede groep archiveringsinstanties gebruikt gaan worden, wat het delen van ontwikkelingskosten en het op lange termijn onderhouden van de gemaakte oplossingen gemakkelijker maakt.

Uiteraard is het goed mogelijk om eigen applicaties te ontwikkelen die dezelfde functionaliteit bevatten. Het grote gevaar is hier echter dat deze oplossing niet makkelijk met oplossingen van anderen kan werken en dat er niet later gemakkelijk emulatoren van derden bij geplaatst kunnen worden, wat in een internationaal framework wel zou werken.

7.2 Nog even wachten

Hieraan gekoppeld is ook het advies om, terwijl er ingezet wordt op bijdrage aan internationale projecten, in de aankomende paar jaren nog niet een emulatiearchief op te zetten waarvan niet zeker is dat het binnen een internationaal framework gaat passen. Over een aantal jaren zijn projecten zoals KEEP (februari 2012) afgerond en is er een brede basis waarmee emulatie als strategie mogelijk wordt. Een mogelijke uitzondering hierop vormt het OVF-formaat.

7.2.1 OVF als uitzondering

Dankzij OVF is er een standaard voor het opslaan van virtuele machines. Zodra er serieuze plannen zijn voor het maken van ondersteuning voor OVF binnen een internationaal framework, kan er gekozen worden om als proef huidige an bijna verouderde systemen te gaan emuleren op virtuele machines in OVF. Ondanks dat deze standaard niet eeuwig zal duren, zorgt ondersteuning binnen internationaal framework er wel voor dat het altijd te openen zal blijven.

7.3 Open Access

Bij het archiveringsvraagstuk wordt vooral rekening gehouden met het blijvend opslaan van gegevens, en het waarborgen hiervan. Minder aandacht is er echter voor het breed toegankelijk maken van de opgeslagen gegevens. Het GRATE project heeft een eerste versie beschikbaar om van een afstand bij emulatoren te kunnen, en wordt nog doorontwikkeld voor gebruik binnen het PLANETS framework. Aangezien voor DANS de toegankelijkheid van data belangrijk is, is het ook zaak om hier genoeg aandacht aan te besteden en actief aan bij te dragen.

7.4 Licenties

Op dit punt is nog meer onderzoek benodigd. Wanneer oude proprietary software in een emulatieomgeving gebruikt blijft worden om gearchiveerde bestanden te kunnen inzien of manipuleren, zal hiervoor een licentie verkregen moeten worden. Wat hier de beschikbare mogelijkheden voor zijn is niet door ons onderzocht en volgend de personen waar we mee gesproken hebben een noemenswaardig probleem. Hier zal binnen DANS en/of binnen andere organisaties nog meer onderzoek naar verricht moeten worden voordat Emulatie als archiveringsstrategie ingezet kan worden op brede schaal. Aan het eind van de zomer zal een paper van Jeffrey van der Hoeven verschijnen over precies dit onderwerp.

Deel II
Implementatie

8 Prototype

Om een idee te geven hoe het beheren van een emulatie-archief kan gebeuren is een prototype beheerapplicatie ontwikkeld. Voor het ontwerp van dit prototype is het goed terug te blikken op de afgelopen secties voor het maken van een geschikt ontwerp.

8.1 Requirements van het Prototype

Ons prototype moet kunnen:

1. Bijhouden welke emulatoren beschikbaar zijn (inclusief op wat voor systemen deze draaien), of welke verouderd raken, (Momenteel in gebruik bij DANS)

2. bijhouden welke data-objecten en applicaties welke emulatoren nodig hebben.

3. Archivarissen/systeembeheerders waarschuwen wanneer support voor een data-object dreigt te vervallen.

   (a) Bestandsformaten die momenteel geëmuleerd worden bij DANS zijn in gevaar om ontoegankelijk te geraken, zodoende zal DANS de vorm van emulatie moeten veranderen of vernieuwen (gebruik makend van View paths).

   Dit zal niet vaak gebeuren, omdat support voor emulatoren alleen veranderd wanneer van server hardware of OS veranderd wordt. Wanneer de server hardware en/of OS veranderd, zal een analyse gedaan moeten worden om te controleren of het nieuwe systeem de huidige applicaties (en emulatoren) ondersteund.

4. Aangeven welke software of operating system van vitaal belang is voor het preserveren van een grote hoeveelheid data-objecten.

8.1.1 Prototype I/O

Het prototype zal op basis van een aantal inputs een bepaalde output moeten kunnen leveren. De specificaties van de output zijn overgenomen uit de stage-opdracht. De inputs en externe projecten vloeiden voort uit het verrichte onderzoek.

1. Inputs:
    - Data-objecten
    - Database van software die we nu emuleren
    - Migratiemogelijkheden
    - Database van Emulatie-omgevingen
2. Gebruikte externe projecten
    - KEEP
    - PRONOM
3. Output van applicatie.
    - Bijhouden welke emulatoren beschikbaar zijn (met de ondersteunde platformen), of welke verouderd raken,
    - welke dataformaten en welke applicaties welke emulatoren vereisen,
    - waarschuw archiveringspersoneel wanneer de ondersteuning voor formaten of applicaties dreigt te verdwijnen.

8.2 Ontwerp

Bij het ontwerp van het prototype is gezocht naar een goede invulling van de hierboven besproken requirements door middel van de bovenstaande inputs en outputs. Hierbij zijn enkele randvoorwaarden opgesteld die voortvloeien uit het gedane onderzoek of die dienen om het binnen de gestelde tijd mogelijk te maken.

8.2.1 Randvoorwaarden

Ten eerste is gebleken dat 'stacked emulation' onhandig is en een flinke performance loss oplevert. Hierdoor hebben we gekozen voor een ontwerp waarin Emulators als basis dienen en zelf niet geïnstalleerd kunnen worden op al geëmuleerde omgevingen. Daarnaast is er voor de ViewPath structuur gekozen om geschiktheid van Emulators en de ondersteunbaarheid van DataObjecten te testen, wat ook terug te zien valt in de Requirements structuur. Aan de hand van de wijzigingen in view paths valt af te leiden hoe goed het gaat met de ondersteuning

van een gearchiveerd bestand. Voor de Requirements zijn nog enkele andere afwegingen gemaakt. Het bijhouden welke Chipsets en Drivers er benodigd zijn voor een correcte ondersteuning bleek niet haalbaar in de gegeven tijd, en er is voor gekozen dit niet mee te nemen in de redenering. Voor Codecs is besloten ze wel mee te nemen, maar er wordt vanuit gegaan dat een Codec voor meerdere Operating Systems hetzelfde is en dat als er voor het archiveren van een DataObject zowel een Codec als een stuk Software nodig is, het stuk Software altijd de Codec kan gebruiken.

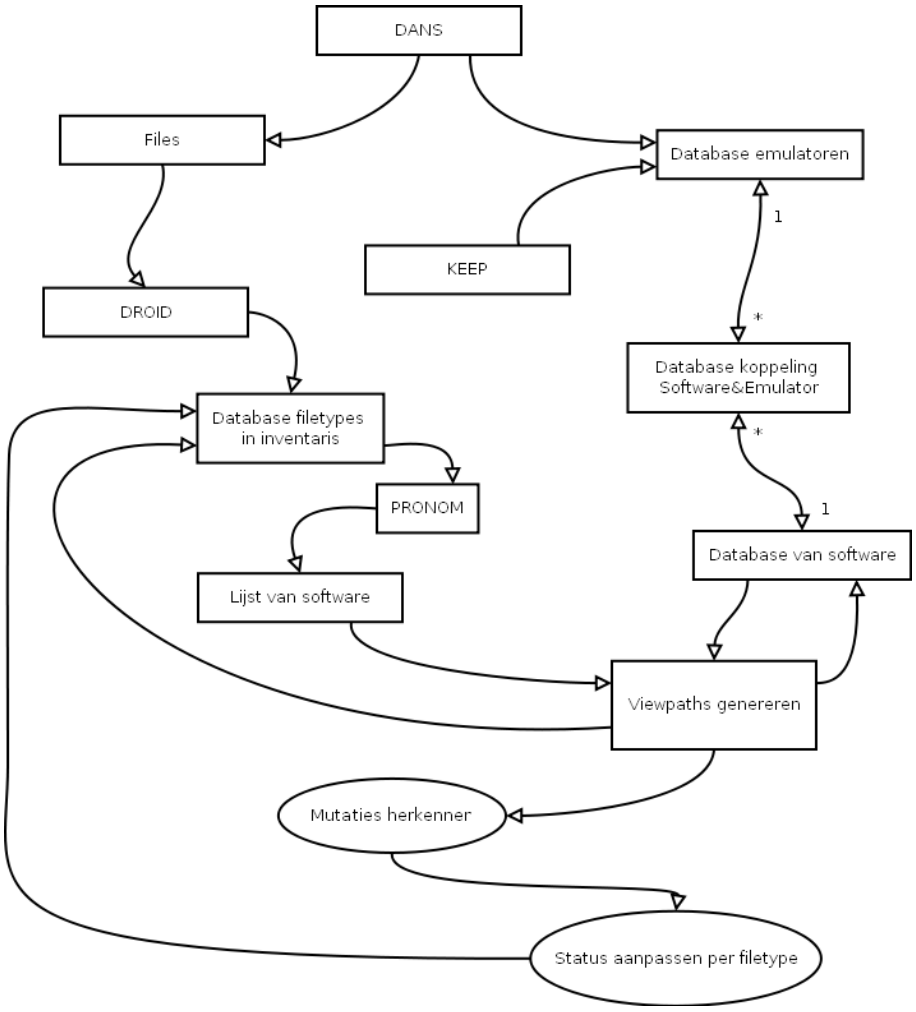
8.2.2 De workflow in het prototype

In figuur 4 is een schema van het systeem waarbinnen ons Prototype komt te staan weergegeven. Een aantal van de blokken zijn externe bronnen, een aantal zijn 'triviaal' (een database bijhouden is simpelweg niet moeilijk). Ons prototype zal dan zorgen voor de koppelingen tussen de blokken en de niet-triviale zaken die met ellipsen weergegeven zijn. Er valt te zien dat vanuit DANS een database van emulatoren en data-objecten gevuld wordt, volgens planning ondersteund door KEEP en DROID. Het prototype zal daarmee aan de hand van deze database viewpaths kunnen genereren, en mutaties kunnen herkennen in de viewpaths, waardoor de status van dataobjecten aangepast kan worden. De status van ieder dataobject reflecteert of er actie ondernomen moet worden om het object te kunnen blijven archiveren. Deze informatie komt weer terug in de database, die ondertussen weer verder gevuld kan worden met nieuwe ondersteunende software, emulatoren etcetera, waardoor de evaluatie opnieuw plaats kan vinden.

8.2.3 Interne werking

De werking van het prototype, zoals al te zien valt in figuur 4, is hieronder in iets meer detail gespecificeerd.

1. Redenatie in applicatie:
   (a) Hou de data-objecten bij, en een lijst welke software gebruikt kan worden voor het bekijken/uitvoeren van deze objecten.
      - Update deze lijst regelmatig door gebruik te maken van PRONOM.
   (b) Cross-reference de lijst van software met de lijst van software die beschikbaar is in een emulatie.
   (c) (Bereken en) Sla per data-object op hoeveel view paths er zijn, er recent maximaal geweest zijn en in totaal maximaal geweest zijn.
   (d) Hou een database met emulatie-omgevingen bij, en de dependencies.
   (e) Als er een emulatie-omgeving wegvalt, erbij komt, of software wegvalt, update de view paths en trek aan de hand daarvan conclusies over de status van data-objecten.
      - Een aantal statussen, die aangeven of er actie benodigd is voor het behouden van deze data.

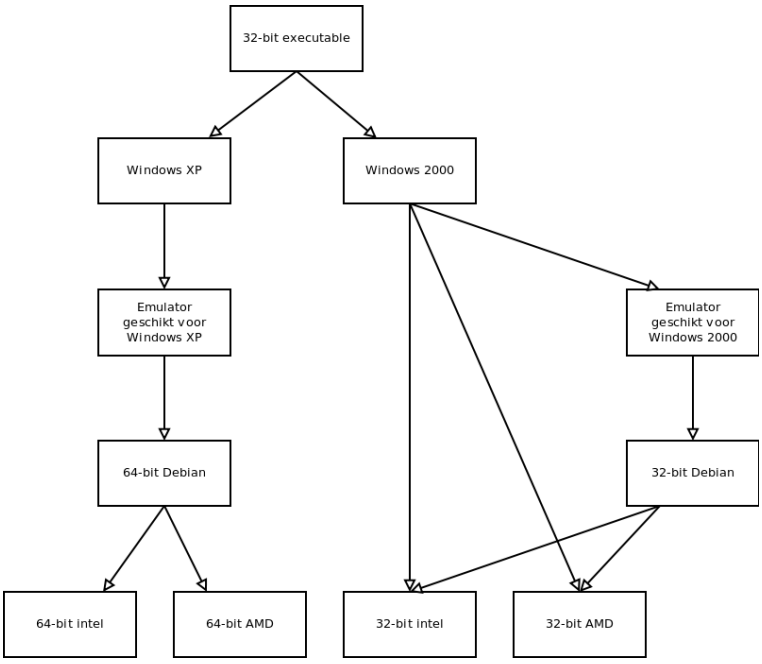


Figuur 4: Schema van ons prototype

- Groeiend aantal viewpaths is goed.
- Grofweg gelijkblijvende hoeveelheid viewpaths is nog geen reden voor paniek
- Een afname van viewpaths (uitgedrukt in procenten vanaf het recente maximum) is reden voor aandacht.
- Een afname boven een bepaalde threshold geeft aan dat er actie ondernomen moet worden.
- Een te laag absoluut aantal view paths geeft aan dat er actie ondernomen moet worden.

Hierbij kan een opsplitsing gemaakt worden tussen view paths die er 'in de wereld' zijn, op basis van PRONOM, en de view paths die DANS heeft. Het eerste kan aangeven wanneer een formaat uit de running raakt en dus een archiefoplossing gevonden moet worden. Het tweede kan aangeven dat het archief in gevaar komt en dat DANS daarop actie moet ondernemen.

Voorbeeld Onderstaand voorbeeld dient om het bovenstaande nog verder toe te lichten. Hier wordt gekeken naar de situatie dat een 32-bits .exe gearchiveerd moet worden. Van deze executable is bekend dat hij op ieder 32-bits Operating System van Microsoft kan werken.



Figuur 5: View paths

View paths (voorbeeld volgens figuur 5):

- WINXP, draait op Emulator WINXP, draait op 64bit debian-based.
- WIN2k, draait op Emulator WIN2k, draait op 32bit debian-based
- WIN2k, draait op 32-bit intel
- WIN2k, draait op 32-bit AMD

Conclusie: de ondersteuning voor 32bit.exe is op dit moment goed.

1. **Beginsituatie:**
   - Recent maximum: 6 view paths.
   - Totaal maximum: 6 view paths.
   - Event: 32-bit debian-based raakt verouderd.
     (a) Gevolg: Emulator WIN2k dreigt ontoegankelijk te worden.
     (b) Gevolg: 32-bits .exe heeft recent 33% van zijn view paths verloren.
     (c) Gevolg: ondersteuning voor 32-bits .exe is **in gevaar**.
2. **Niet al te lang hierna:**
   - Recent maximum: 6 view paths.
   - Totaal maximum: 6 view paths
   - Event: 32-bit AMD raakt verouderd.
     (a) Gevolg: WIN2k dreigt ontoegankelijk te worden.
     (b) Gevolg: 32-bits .exe heeft recent 50% van zijn view paths verloren.
     (c) Gevolg: 32-bits .exe heeft **aandacht** nodig.
     (d) Gevolg: 32-bits .exe heeft slechts 3 view paths over, dat is te weinig.
     (e) Gevolg: 32-bits .exe heeft **direct aandacht** nodig.
3. **Actie ondernomen:**
   - Event: DANS maakt nieuw viewpath.
     (a) Gevolg: WIN2k beter toegankelijk
     (b) Gevolg: 32-bits .exe heeft nieuwe view paths er bij gekregen.
     (c) Gevolg: Ondersteuning 32-bits .exe is **goed**.
4. **Lang hierna:**
   - Recent maximum: 4 view paths.
   - All-time maximum: 6 view paths.
   - Event: 32-bits intel is niet meer toegankelijk.
     (a) Gevolg: WIN2k dreigt ontoegankelijk te worden.

(b) Gevolg: 32-bits .exe heeft recent 25% van zijn view paths verloren.

(c) Gevolg: ondersteuning voor 32-bits .exe is **in gevaar**.

(d) Gevolg: 32-bits .exe heeft slechts 3 view paths over, dat is te weinig.

(e) Gevolg: 32-bits .exe heeft **direct aandacht** nodig.

8.3 Ontwikkelmethode

Aan het begin van het ontwikkeltraject moest er een keuze gemaakt worden op welke manier we het prototype gingen implementeren, en welke teststrategie er aangehouden ging worden. Omdat het door de complexiteit van het product en het nog lopende onderzoek te verwachten was dat we meer methodes nodig gingen hebben dan oorspronkelijk bedacht, is er gekozen voor een agile programmeervariant waarbij alle methoden geschreven gingen worden die we op dat moment al nodig hadden, en nieuwe functies toegevoegd zouden worden wanneer en waarvoor dat nodig zou blijken.

8.3.1 Testmethode

Een testmethode die hierbij paste was JUnit Testing, waarbij we elke class afzonderlijk getest hebben, en de losse methoden. Dit hebben we in het begin van het implementatietraject opgepakt en uitgevoerd, waarmee we ook de nodige kleine bugs uit het prototype gehaald hebben. Vanwege het doel van het prototype was echter niet alles van belang om te testen, zoals zal blijken uit de emma coverage report.

8.3.2 Coverage testing met Emma

Bij het Unit testen met JUnit is gekozen om de code coverage bij te houden met EclEmma, een Emma-implementatie voor Eclipse. De uitvoer hiervan is te zien in bijlage B.

Ten eerste moet hierbij de observatie geplaatst worden dat de belangrijkste functionaliteit van het prototype zich bevind in de ArchiveManager als redeneer-centrum van de applicatie, en in de verschillende reports en SubRequirements, die de basis van de emulatiestructuur vormen. De SAME_Database dient slechts als facilitator met dezelfde vijf standaardmethoden voor elke lijst die we erin bijhouden: add, set, get, has, remove. Voor elk van de tien lijsten die in SAME_Database worden bijgehouden zijn deze vijf methoden geimplementeerd op precies dezelfde manier, met enkel een variabelenaam als verschil. Dit maakt deze methoden zo triviaal dat er slechts een getest hoeft te worden om ze allemaal te controleren, waardoor we ervoor gekozen hebben om ze niet allemaal expliciet in JUnit tests te controleren.

Bij een aantal andere classes zien we dat er slechts 60% coverage is. Dit zijn met name de klasses uit het protodans.data package. De niet gecoverde onderdelen zijn onder andere een aantal ongebruikte constructoren. Deze constructoren hebben we, net als bij de database-methoden hierboven, erin gezet voor het geval dat iemand ze nodig heeft, terwijl ze in onze applicatie verder niet gebruikt worden. Aangezien deze constructoren slechts triviale code bevatten achtten we het niet van belang deze expliciet te testen.

Naast deze classes geeft ook het protodans.external package een lage coverage. In dat package zit de koppeling van onze applicatie met de externe projecten DROID en PRONOM. De koppeling met DROID wordt getest, maar krijgt een lage coverage doordat een van de constructoren niet gebruikt wordt. Die constructor is echter wel zo triviaal dat het niet testen ervan niet erg is.
De koppeling met PRONOM is hiernaast ook nooit afgemaakt, omdat PRONOM geen API biedt en het parsen van de website van PRONOM voor de informatie die in ons prototype gebruikt wordt ten eerste lastig bleek en ten tweede niet een hoofddoel van dit prototype was. Omdat deze koppeling niet gebruikt wordt in het prototype is het ook niet nodig deze te testen.

Tot slot is er totaal geen coverage in het gui package. Voor Unit testing is dit echter volkomen terecht, de GUI wordt namelijk alleen maar gebruikt om de applicatie grafisch weer te geven en niet om de nodige redenaties te verrichten.

8.4 Design Decisions

Bij het maken van het klasse-model in appendix A zijn een aantal beslissingen genomen.
Allereerst: het DataObject (figuur 8). Over een gearchiveerd data-object (zij het software, zij het andere bestandsformaten) zal een aantal gegevens beschikbaar moeten blijven.

Onder deze gegevens valt ook wat er benodigd is om een data-object toegankelijk te houden. Dit wordt met Requirements bijgehouden.
Er is gekozen om verschillende type afhankelijkheden te kunnen onderscheiden, te zien in figuur 11:

- Software
- Operating System
- Codec(s)
- Chipset
- Driver

De laatste twee van deze typen zullen verder redelijk buiten beschouwing gelaten worden. Hun isAvailable method zal dan ook simpelweg 'true' retourneren.

Naast de verschillende type afhankelijkheden wordt bijgehouden of een afhankelijkheid enkel- of meervoudig is, of zelfs 'incompatible'. Dit wordt aangegeven door op te slaan of er (tenminste) één, allemaal, of zelfs geen (incompatibiliteit) van de aangegeven afhankelijkheden beschikbaar dient te zijn. Een data-object kan meerdere van die sets van afhankelijkheden hebben. Zo zou een mediabestand afhankelijk kunnen zijn van een codec uit de H.264 familie en een of andere mediaplayer, maar uit beide sets is het weer niet belangrijk welke gebruikt wordt.
Voor elke Requirement wordt bijgehouden in een database of deze bij DANS beschikbaar is. Dat is dan ook op te vragen met isAvailable.
Uit deze Requirements kunnen ViewPaths (figuur 12) gebouwd worden. ViewPaths zijn handig om, zodra uitgerekend, los te bewaren. Hiermee kan een hoop interessante informatie snel toegankelijk gehouden worden. Zo is het bijvoorbeeld mogelijk om het effect, over de gehele linie van gearchiveerde data-objecten bij DANS, van het wegvallen van een stuk software te bekijken.

Emulator Hoe de emulator in ons systeem komt te staan is als volgt: Een emulator kan gezien worden als *vervanger* van een Operating System. Het is hiervoor gebouwd op een specifieke chipset. Vanwege de tijdsbeperkingen van het bachelorproject is ervoor gekozen de Chipset hierin wel als mogelijkheid op te nemen, maar niet te implementeren. Er wordt in het prototype dus aangenomen dat een specifiek OS op een willekeurige chipset geinstalleerd kan worden. Dit is te zien in figuur 13.

Systeembeheerder De output van de beheerapplicatie zal in veel gevallen een opdracht voor een systeembeheerder opleveren. Dit kunnen een aantal dingen zijn.

1. Een stuk software moet ter archivering op de juiste emulator toegevoegd worden;
2. Een specifieke codec of driver moet op een specifieke emulator toegevoegd worden;
3. Een softwarepakket moet op een specifieke emulator geïnstalleerd worden, om het archiveren van een DataObject te faciliteren;
4. Er moet een nieuw OS geëmuleerd gaan worden.

9 Bevindingen door prototype

Naar aanleiding van het maken ontwerpen, ontwikkelen en demonstreren van dit prototype, zijn een aantal zaken aanzienlijk duidelijker geworden.

Voornamelijk is duidelijk geworden dat het beheren van emulatieomgevingen door tools als DROID, PRONOM, JHOVE, KEEP, en een systeem gelijkend aan

het ontwikkelde prototype vergemakkelijkt wordt. Deze vergemakkeling zorgt ervoor dat Emulatie als archiveringsmethode geen herculeïsche taak blijft maar een haalbare mogelijkheid wordt, doordat het met behulp van deze genoemde tools niet buiten proportie met migratie zal staan in termen van manuren en algemene kosten. Door de beslissingen over welke emulatoren te gebruiken met welke voorgeïnstalleerde software over te laten kan een systeembeheerder zich focussen op zijn echte taak: systemen beheren.

Tevens is duidelijk geworden hoe belangrijk het is om samen te werken met andere (internationale) projecten op dit gebied. Het bijhouden van de vereisten voor alle bestandstypen en van alle softwarepakketten (en wanneer de softwarepakketten niet meer ondersteund worden) is een herculeïsche taak. Door in samenwerkingsverband bij te houden welke softwaren op welk systeem draait en welke software welke bestandsformaten kan ondersteunen, wordt dit echter mogelijk. In PRONOM leek hiervoor een oplossing gevonden, het automatisch gebruik hiervan bleek echter tegen te vallen door het ontbreken van een API.

Hiernaast blijkt het archiveren van software met emulatie niet complexer te zijn dan het archiveren van statische objecten met emulatie, wat emulatie een generieke maar alsnog moeilijke optie maakt voor alle objecten die gearchiveerd moeten worden.

Door het prototype is ook het inzicht ontstaan dat een efficiente implementatie van een uiteindelijke beheeromgeving ten zeerste van belang is. In de implementatie van het besproken prototype is niet afdoende rekening gehouden met de efficientie van algoritmen voor het berekenen van ondersteuning, om een grote database van software, emulatoren en te archiveren dataobjecten te kunnen ondersteunen. In een applicatie die uiteindelijk gebruikt zal gaan worden voor het beheren van emulatieomgevingen, mogelijk binnen de Open Planets Foundation, zal hier een stuk meer aandacht aan gegeven moeten worden.

Referenties

[1] Data Archiving and Networked Services, "Over DANS." http://dans.knaw.nl/content/over-dans, June 2010.

[2] E. Oltmans and N. Kol, "A comparison between migration and emulation in terms of costs," *RLG DigiNews*, vol. 9, April 2005. http://worldcat.org/arcviewer/1/OCC/2007/07/10/0000068902/viewer/file1.html.

[3] D. Roorda, "Migration to Intermediate XML for Electronic Data (MIXED)," 2007. http://mixed.dans.knaw.nl/files/file/white_paper2.pdf.

[4] J. van der Hoeven *et al.*, "Emulation for digital preservation in practice: The results," *The International Journal of Digital Curation*, vol. 2, pp. 123–131, December 2007. http://www.ijdc.net/index.php/ijdc/article/viewArticle/50.

[5] R. Lorie, "Long term preservation of digital information," in *Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pp. 346–352, ACM, 2001. http://doi.acm.org/10.1145/379437.379726.

[6] J. Rothenberg, "Ensuring the longevity of digital information," *Scientific American*, vol. 272, pp. 42–7, January 1995. Revision of February 22nd, 1999.

[7] H. van Wijngaarden and E. Oltmans, "Digital preservation and permanent access: The uvc for images," tech. rep., Koninklijke Bibliotheek, National Library of the Netherlands, 2004.

[8] J. van der Hoeven, "Uvc and emulation as preservation strategies." http://www.kb.nl/hrd/dd/dd_onderzoek/20050415_uvc_emulation_lund_sweden_jrvanderhoeven.ppt, April 2005.

[9] Koninklijke Bibliotheek and Nationaal Archief, "Dioscuri, the durable x86 emulator in java." http://dioscuri.sourceforge.net.

[10] European Unions Seventh Framework Programme, "Keeping emulation environments portable." http://www.keep-project.eu/.

[11] I. VMWare, Inc. en XenSsrouce, "The Open Virtual Machine Format, Whitepaper for OVF Specification," 2007. http://www.vmware.com/pdf/ovf_whitepaper_specification.pdf.

[12] VMWare, "Open virtualization format." http://www.vmware.com/appliances/getting-started/learn/ovf.html.

[13] European Unions Sixth Framework Programme, "The planets project." http://www.planets-project.eu/.

[14] Open Planets Foundation, "Open planets foundation." http://www.openplanetsfoundation.org/.

[15] The National Archives, "Digital record object identification." http://droid.sourceforge.net.

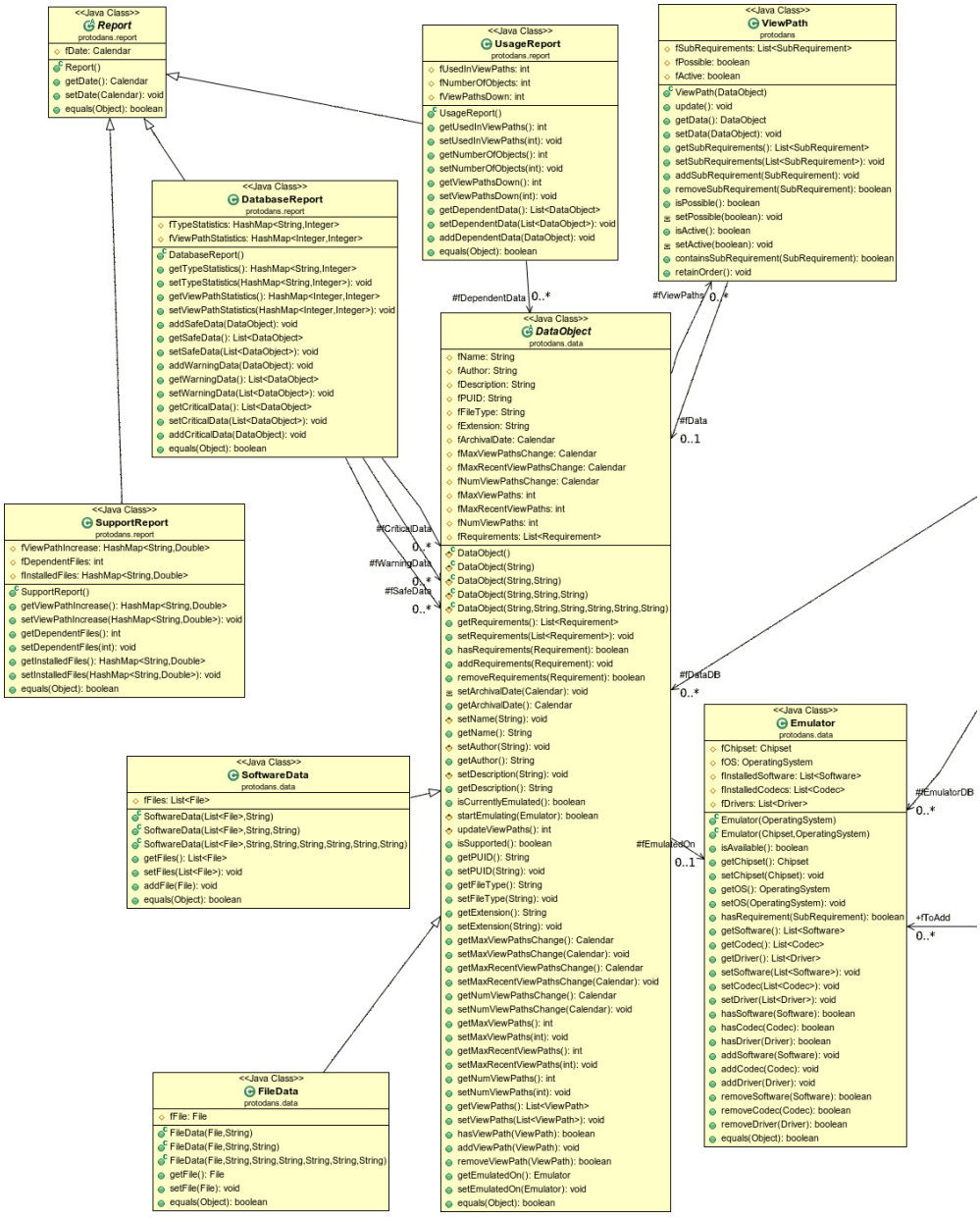
[16] JSTOR and the Harvard University Library, "Jstor/harvard object validation environment." http://hul.harvard.edu/jhove/.

[17] The National Archives, "Pronom online registry of technical information." http://www.nationalarchives.gov.uk/PRONOM/Default.aspx.

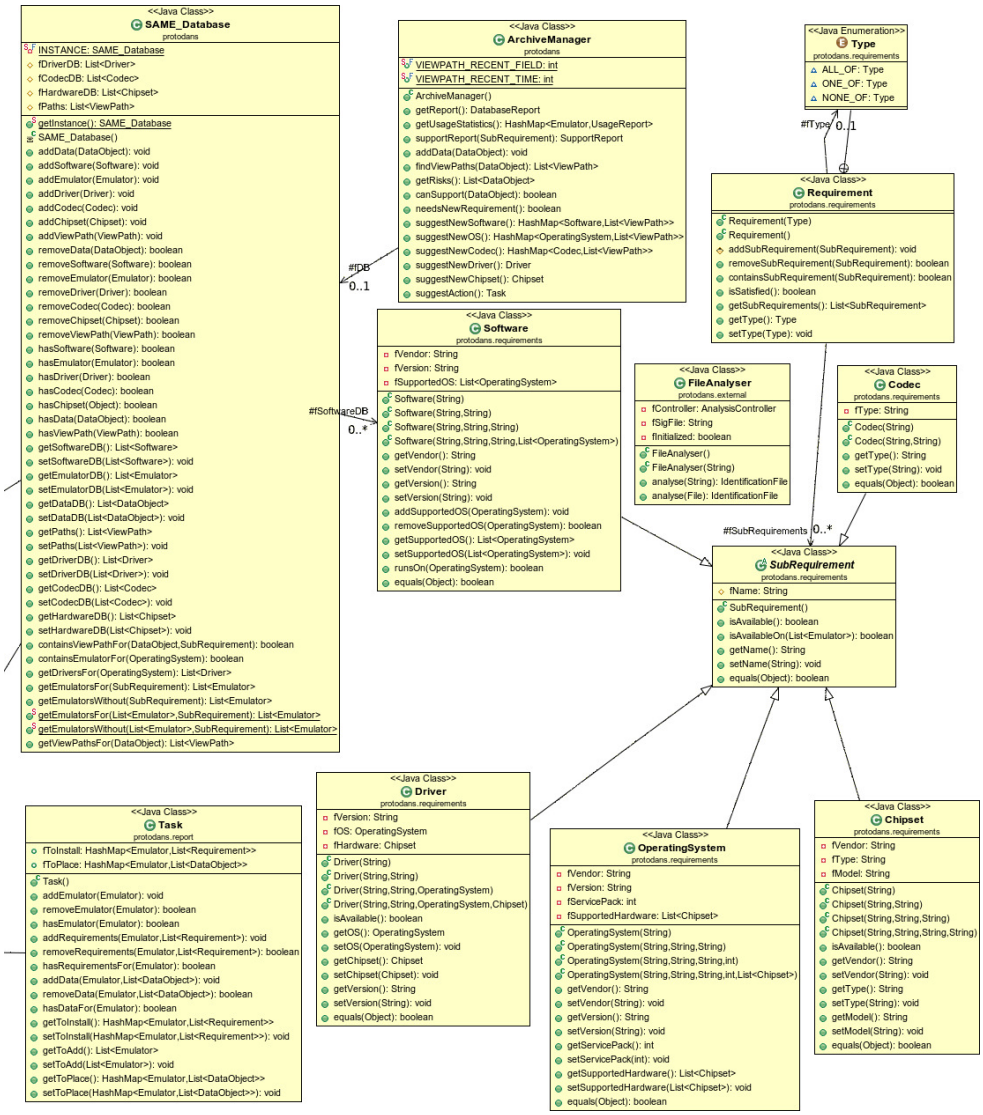
[18] D. von Suchodoletz and J. van der Hoeven, "Emulation: From digital artefact to remotely rendered environments," *The International Journal of Digital Curation*, vol. 4, pp. 146–155, December 2009. http://www.ijdc.net/index.php/ijdc/article/viewArticle/50.

Deel III
Bijlagen

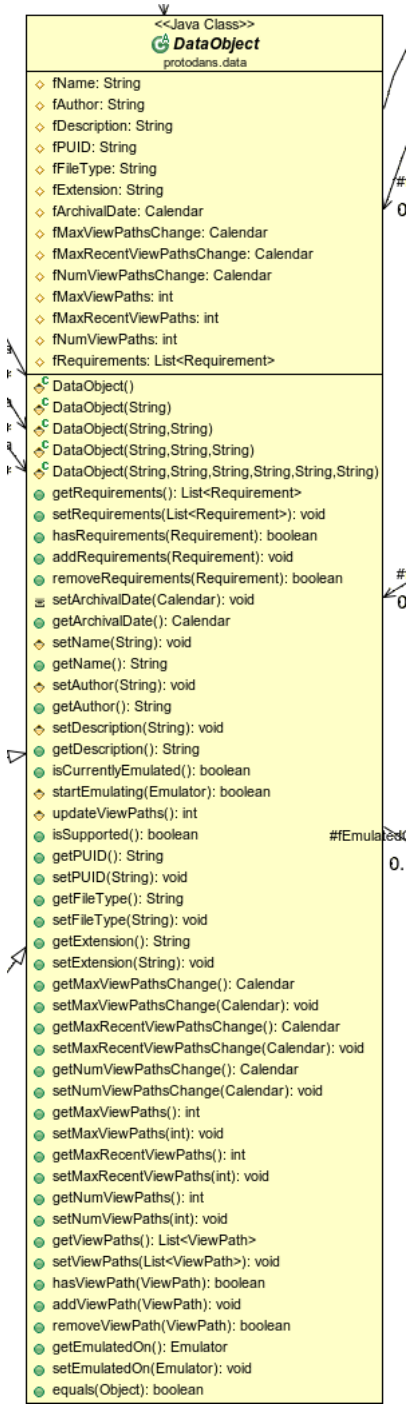
A Prototype Class Diagrams



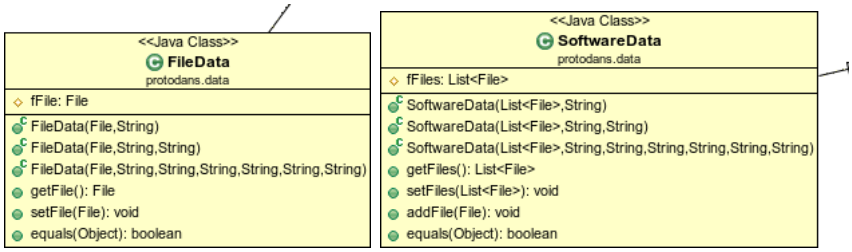
Figuur 6: Het volledige klassendiagram (deel 1)



Figuur 7: Het volledige klassendiagram (deel 2)

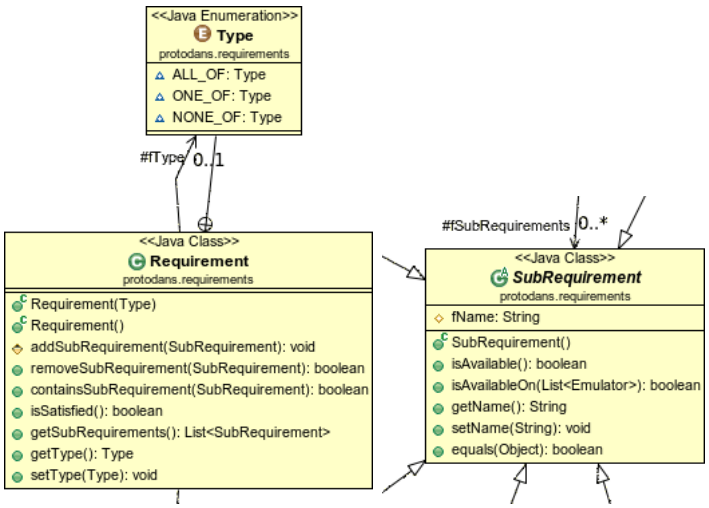


Figuur 8: Het DataObject



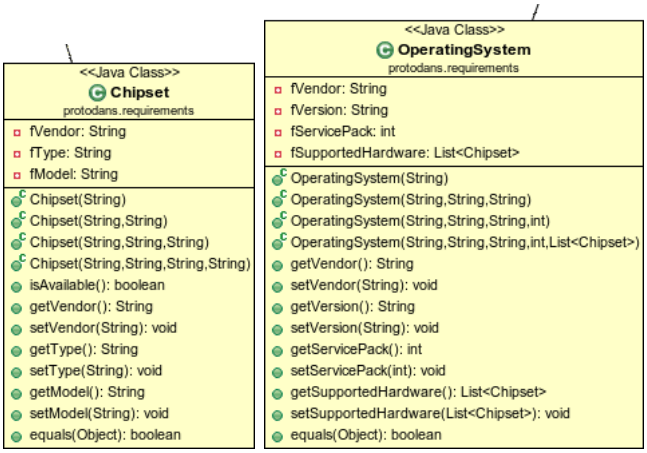
(a) Gewone data (b) Software

Figuur 9: Verschillende typen data-objecten



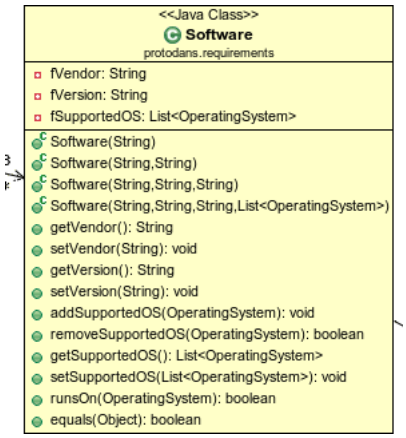
(a) Een wrapper om een tupel SubRequirements (b) Een enkel benodigd stuk soft- of hardware

Figuur 10: Een framework om requirements in weer te geven

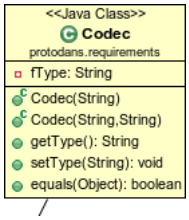


(a) Chipset

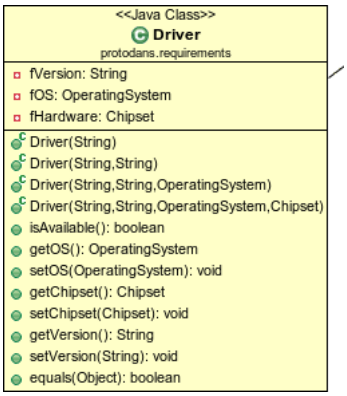
(b) Operating System



(c) Software

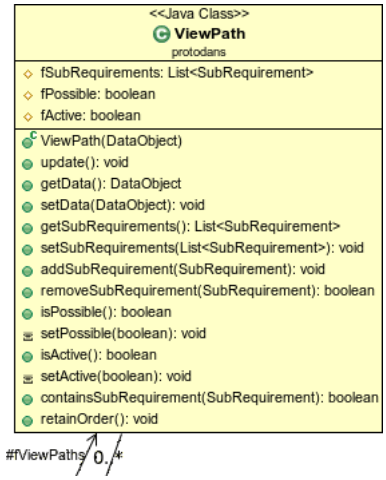


(d) Codec

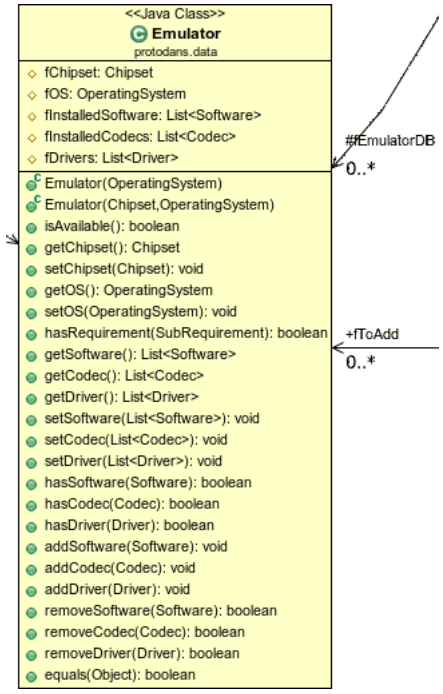


(e) Driver

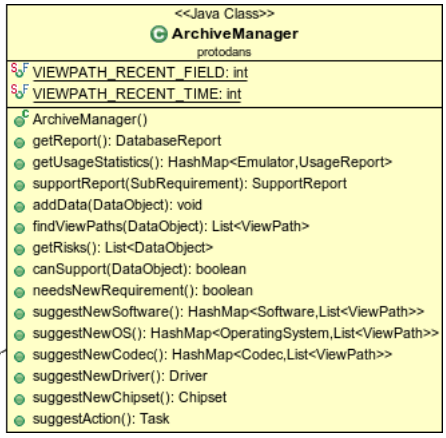
Figuur 11: De verschillende SubRequirements



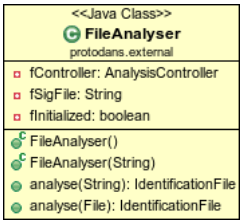
Figuur 12: De representatie van een ViewPath



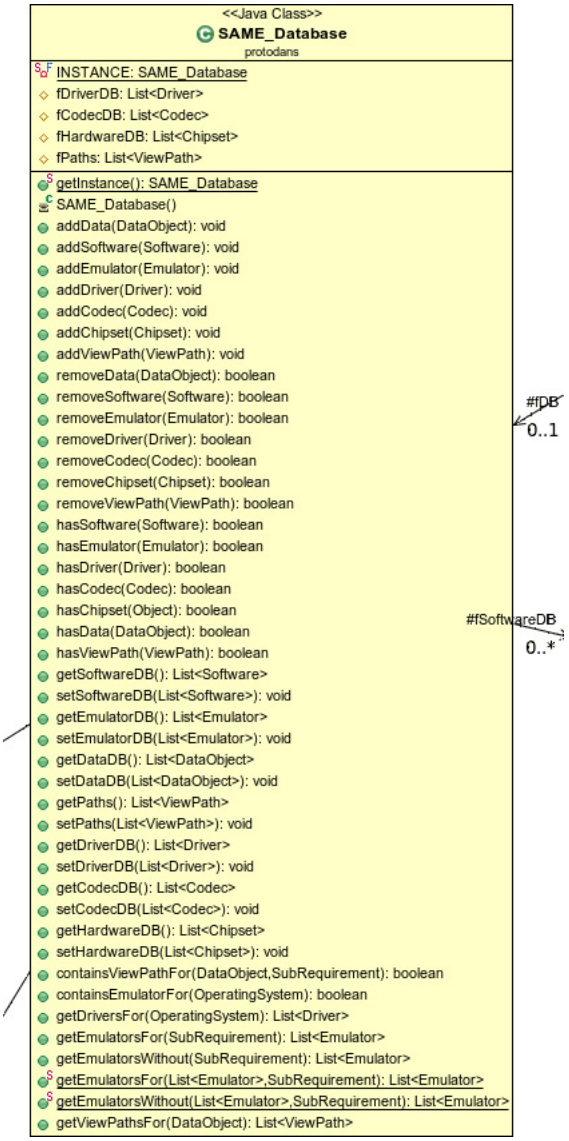
Figuur 13: De Emulator klasse



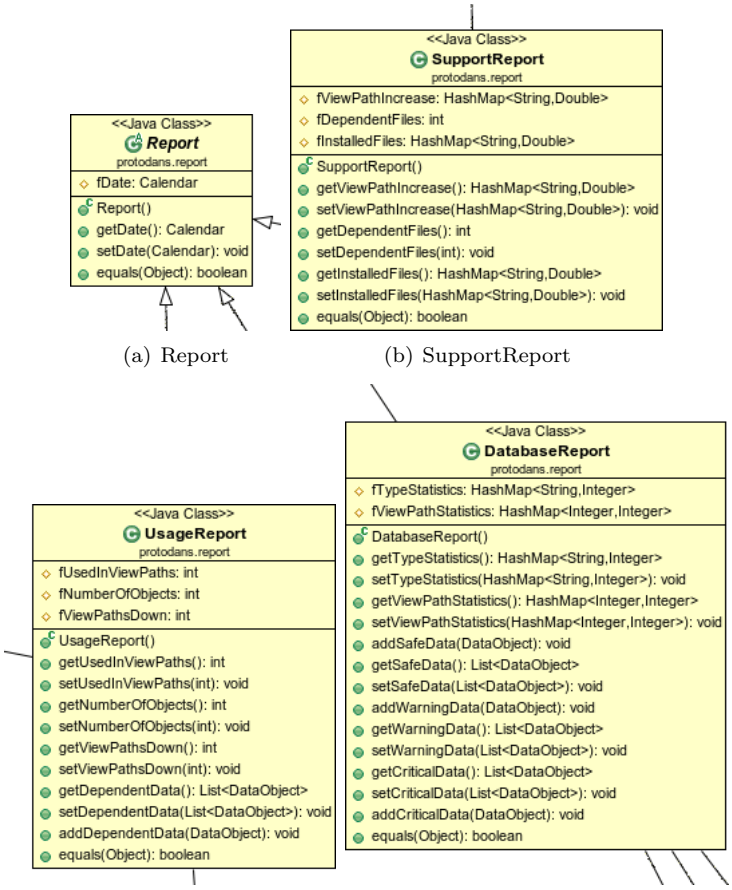
Figuur 14: De klasse die het toevoegen/beredeneren van en over data-objecten regelt



Figuur 15: Klasse voor analyse van bestanden



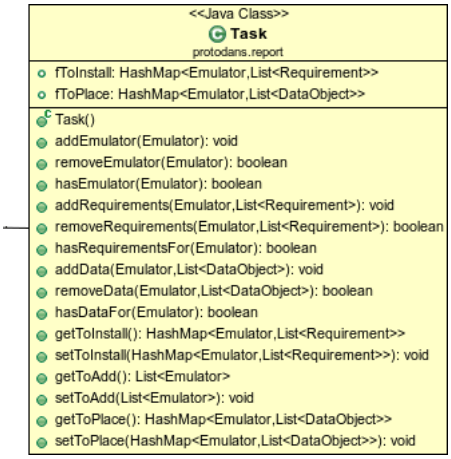
Figuur 16: De database die alle objecten in nette lijsten bijhoudt



(a) Report (b) SupportReport

(c) UsageReport (d) DatabaseReport

Figuur 17: De verschillende Reports die gegenereerd kunnen worden



Figuur 18: Een taak voor de systeembeheerder

B Emma Coverage Report

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

[all classes]

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %	line, %
all classes	61% (45/74)	60% (340/568)	53% (11391/21441)	54% (2469.5/4576)

OVERALL STATS SUMMARY

total packages:	12
total executable files:	69
total classes:	74
total methods:	568
total executable lines:	4576

COVERAGE BREAKDOWN BY PACKAGE

name	class, %	method, %	block, %	line, %
protodans.gui	0% (0/27)	0% (0/128)	0% (0/8144)	0% (0/1675)
protodans.external	33% (1/3)	22% (2/9)	7% (29/420)	7% (6.4/86)
protodans.data	100% (4/4)	60% (58/97)	69% (1005/1453)	64% (221.7/346)
protodans	100% (3/3)	64% (67/105)	74% (2076/2810)	72% (456.5/633)
protodans.requirements	100% (9/9)	87% (72/83)	84% (892/1056)	88% (201.6/228)
protodans.report	100% (5/5)	100% (53/53)	94% (719/768)	96% (153.9/160)
test.protodans.report	100% (5/5)	100% (20/20)	96% (1431/1485)	99% (241.4/244)
test	100% (3/3)	67% (10/15)	98% (2861/2908)	98% (657.9/673)
test.protodans	100% (3/3)	100% (12/12)	99% (635/641)	100% (158.6/159)
test.protodans.requirements	100% (7/7)	100% (28/28)	99% (1635/1648)	100% (328.5/329)
test.protodans.data	100% (4/4)	100% (16/16)	100% (99/99)	100% (38/38)
test.protodans.external	100% (1/1)	100% (2/2)	100% (9/9)	100% (5/5)

[all classes]

EMMA 2.0.5312 EclEmma Fix 1 (C) Vladimir Roubtsov

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

all classes

COVERAGE SUMMARY FOR PACKAGE [protodans]

name	class, %	method, %	block, %	line, %
protodans	100% (3/3	64% (67/105	74% (2076/2810	72% (456.5/633

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
SAMEDatabase.java	100% (1/1	53% (37/70	49% (448/909	50% (109.6/221
ArchiveManager.java	100% (1/1	73% (11/15	83% (1219/1465	80% (240.4/299
ViewPath.java	100% (1/1	95% (19/20	94% (409/436	94% (106.5/113

all classes

EMMA 2.0.5312 (EclEmma Fix 1) (C) Vladimir Roubtsov

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

all classes

COVERAGE SUMMARY FOR PACKAGE [protodans.data]

name	class, %	method, %	block, %	line, %
protodans.data	100% (4/4	60% (58/97	69% (1005/1453	64% (221.7/346

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
Emulator.java	100% (1/1	40% (10/25	50% (157/312	44% (33.2/76
SoftwareData.java	100% (1/1	60% (6/10	61% (58/95	53% (13.8/26
FileData.java	100% (1/1	67% (4/6	64% (43/67	58% (9.8/17
DataObject.java	100% (1/1	68% (38/56	76% (747/979	73% (164.9/227

all classes

EMMA 2.0.5312 EclEmma Fix 1 (C) Vladimir Roubtsov

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

all classes

COVERAGE SUMMARY FOR PACKAGE [protodans.external]

name	class, %	method, %	block, %	line, %
protodans.externa.	33% (1/3	22% (2/9	7% (29/420	7% (6.4/86

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
HelloFiletype.jav	0% (0/1	0% (0/2	0% (0/193	0% (0/27
PronomXMLoader.jav	0% (0/1	0% (0/3	0% (0/174	0% (0/44
FileAnalyser.jav	100% (1/1	50% (2/4	55% (29/53	43% (6.4/15

all classes

EMMA 2.0.5312 EclEmma Fix : (C) Vladimir Roubtsov

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

all classes

COVERAGE SUMMARY FOR PACKAGE [protodans.report]

name	class, %	method, %	block, %	line, %
protodans.repor	100% (5/5	100% (53/53	94% (719/768	96% (153.9/160

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
Report.java	100% (1/1	100% (4/4	88% (35/40	89% (8.9/10
SupportReport.java	100% (1/1	100% (9/9	91% (117/129	93% (23.2/25
DatabaseReport.java	100% (1/1	100% (15/15	91% (230/252	97% (43.6/45
UsageReport.java	100% (1/1	100% (9/9	94% (94/100	97% (24.3/25
Task.java	100% (1/1	100% (16/16	98% (243/247	98% (54/55

all classes

EMMA 2.0.5312 EclEmma Fix 1 (C) Vladimir Roubtsov

EMMA Coverage Report (generated Fri Jul 23 11:33:31 CEST 2010)

all classes

COVERAGE SUMMARY FOR PACKAGE [protodans.requirements]

name	class, %	method, %	block, %	line, %
protodans.requirements	100% (9/9	87% (72/83	84% (892/1056	88% (201.6/228

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
SubRequirementComparator.java	100% (1/1	67% (2/3	66% (73/110	64% (12.8/20
Requirement.java	100% (2/2	80% (12/15	75% (217/289	84% (49.8/59
Software.java	100% (1/1	80% (12/15	83% (125/151	81% (26/32
OperatingSystem.java	100% (1/1	93% (13/14	85% (123/145	97% (32/33
SubRequirement.java	100% (1/1	86% (6/7	95% (58/61	93% (14/15
Chipset.java	100% (1/1	92% (11/12	98% (121/123	96% (27/28
Driver.java	100% (1/1	92% (11/12	98% (121/123	96% (27/28
Codec.java	100% (1/1	100% (5/5	100% (54/54	100% (13/13

all classes

EMMA 2.0.5312 (EclEmma Fix 1) (C) Vladimir Roubtsov

C Planning

Weekplanning	Datum	Dagspecificatie
Week 0 – Inlezen	do 06-mei 10	
	vr 07-mei 10	
	za 08-mei 10	
	zo 09-mei 10	
Week 1	ma 10-mei 10	15.30 Afspraak met Peter van Nieuwenhuizen
Bespreken en onderzoeken	di 11-mei 10	Begin rapport, opzet
	wo 12-mei 10	Jeff weg
	do 13-mei 10	Hemelvaartdag
	vr 14-mei 10	DANS gesloten
	za 15-mei 10	
	zo 16-mei 10	
Week 2	ma 17-mei 10	15.00 Afspraak met Jeffrey van der Hoeven @ KB
Afspraak vdHoeven	di 18-mei 10	
Afspraak Roorda	wo 19-mei 10	
Rapport vullen	do 20-mei 10	
	vr 21-mei 10	
	za 22-mei 10	
	zo 23-mei 10	
Week 3	ma 24-mei 10	Tweede pinksterdag
Rapport vullen	di 25-mei 10	Marion vragen naar Groningen, (mogelijke andere usecases)
1e bevindingen	wo 26-mei 10	
Begin ontwerp prototype	do 27-mei 10	
Usecases verzamelen	vr 28-mei 10	15.45 Afspraak met Peter van Nieuwenhuizen
1e bevindingen	za 29-mei 10	
	zo 30-mei 10	
Week 4	ma 31-mei 10	16.00 Afspraak Rudi Usecases
Prototype schema's uitwerken	di 01-jun 10	
	wo 02-jun 10	
	do 03-jun 10	
	vr 04-jun 10	
	za 05-jun 10	
	zo 06-jun 10	
Week 5	ma 07-jun 10	
Prototype schema's uitwerken	di 08-jun 10	

Rapport herzien	wo 09-jun 10	
	do 10-jun 10	15.45 Afspraak met Peter van Nieuwenhuizen
	vr 11-jun 10	
	za 12-jun 10	
	zo 13-jun 10	
Week 6	ma 14-jun 10	
Begin implementatie prototype	di 15-jun 10	15:30 Afspraak Maurice van den Dobbelsteen (Nationaal Archief)
	wo 16-jun 10	
	do 17-jun 10	
	vr 18-jun 10	
	za 19-jun 10	
	zo 20-jun 10	
Week 7	ma 21-jun 10	15.45 Afspraak met Peter van Nieuwenhuizen
Nieuwe bevindingen in rapport	di 22-jun 10	
	wo 23-jun 10	
	do 24-jun 10	
	vr 25-jun 10	
	za 26-jun 10	
	zo 27-jun 10	
Week 8	ma 28-jun 10	
Eerste versie prototype af	di 29-jun 10	
Prototype testen en showen	wo 30-jun 10	
	do 01-jul 10	
	vr 02-jul 10	
	za 03-jul 10	
	zo 04-jul 10	
Week 9	ma 05-jul 10	15.45 Afspraak met Peter van Nieuwenhuizen
Prototype herzien, fixes	di 06-jul 10	
Conclusies trekken, voor rapport	wo 07-jul 10	
	do 08-jul 10	
	vr 09-jul 10	
	za 10-jul 10	
	zo 11-jul 10	
Week 10	ma 12-jul 10	
Prototype af	di 13-jul 10	

Rapport af	wo 14-jul 10	
Prototype showen	do 15-jul 10	
Rapport bespreken.	vr 16-jul 10	
	za 17-jul 10	
	zo 18-jul 10	
Uitloop	ma 19-jul 10	
	di 20-jul 10	
	wo 21-jul 10	
	do 22-jul 10	Presentatie bij DANS
	vr 23-jul 10	