Procesverslag

Ontwikkelen van een synchronisatie tool volgens het IST/SOLL model voor een internet exchange backbone netwerk

Dit document is opgesteld door Gerben Geijteman
in opdracht van de Netherlands Internet Exchange (NL-ix)
19-3-2012

Versie: Final

Referaat

Gerben Geijteman, Afstudeerverslag, Ontwikkelen van een synchronisatie tool volgens het IST/SOLL model voor een internet exchange backbone netwerk, Netherlands Internet Exchange (NL-ix), zondag 19 februari 2012.

Dit verslag is geschreven in het kader van het afstuderen van Gerben Geijteman voor de opleiding Technische Informatica aan de Haagse Hogeschool. Het verslag beschrijft de stageperiode van 20 weken bij Netherlands Internet Exchange (NL-ix) te Den Haag, gedurende de periode van 1 juni 2011 tot 16 december 2011.

Descriptoren:

- Afstudeerverslag
- Backbone
- Database
- Programmeren
- Internet Exchange
- Agile Unified Process

Voorwoord

Dit document is geschreven in het kader van het HBO afstuderen voor Technische Informatica op de academie ICT en Media van de Haagse Hogeschool. De opdracht is uitgevoerd van 1 juni 2011 tot en met 19 december 2011 bij de Netherlands Internet Exchange.

Bij het lezen van dit document word enige kennis vereist over technische infrastructuren. Het helpt als de lezer wegwijs is met netwerkprotocollen. Een verklarende woordenlijst vind u in de bijlagen.

Ik wil graag de volgende mensen bedanken voor hun bijdrage:

- Jan Hoogenboom, voor zijn rol als opdrachtgever en begeleider.
- Gert-Jan Aalderink, voor het meenemen op sleeptouw, de boeken en de pepernoten.
- Rutger Spaans, voor het terugwinnen van mijn vertrouwen in het onderwijs.
- Madelon Nieuwland, voor haar altijd duidelijke feedback.
- Robin Collard, voor zijn initiële inzet in het project.
- Jennifer Kula, voor haar hulp tijdens de nachtelijke sparmomenten.
- Nel Bouwhuis, met spellingscontrole alleen red ik het niet.
- Mijn vrienden en familie voor hun nooit aflatende vertrouwen in mij.

Veel plezier met het lezen.

Gerben Geijteman, Noordwijk, 18 maart 2012

Inhoudsopgave

1 Inleiding

Vanwege de ambitie om de kennis in backbone netwerken te verdiepen ben ik in contact gekomen met de Netherlands Internet Exchange (verder NL-ix). Bij NL-ix werd mij de kans gegeven om mij hier verder in te verdiepen.

Dit verslag beschrijft het keuzeproces, de procesbeheersing en de invulling van de tijd die voor dit project genomen is. Er is tijdens het proces gekozen voor een iteratieve aanpak onder de Agile ontwikkelmethode. Deze keuze heeft de structuur van dit document gevormd, met name van het iteratieve aspect is duidelijk gebruik gemaakt.

Grofweg is het document opgedeeld in drie stukken.

Eerst de opdracht en de invulling van het beheersproces. Hier word verder uitgewijd over de details van het project, de keuzes van de inrichting van het beheersproces en waarom dit anders is dan initieel beschreven in de opdrachtomschrijving.

Het tweede gedeelte bestaat uit de invulling en uitwerking van dit proces. De invulling word gekenmerkt door voornamelijk technische keuzes en de communicatie met de opdrachtgever.

Het derde en laatste gedeelte is de evaluatie. Dit gedeelte beschrijft uiteraard de procesvorming, de producten, de beroepstaken en de evaluatie van het proces.

2 De organisatie en opdracht

2.1 Beschrijving van de organisatie

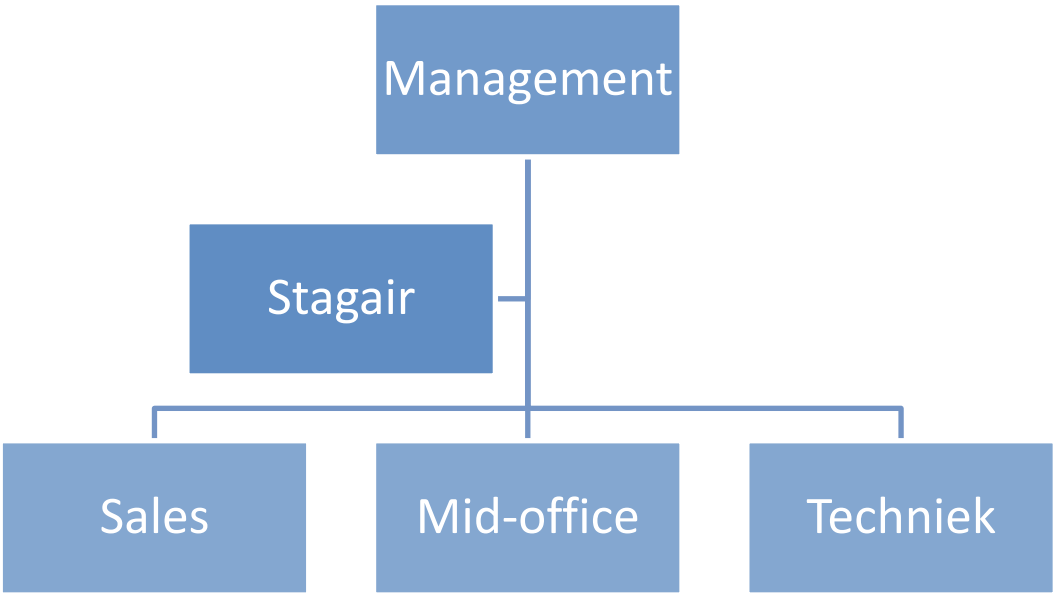
NL-ix is, zoals de naam dat suggereert een Internet Exchange[1]. NL-ix is de merknaam voor het bedrijf Broadband Hosting BV. Tevens opereert men onder de merknamen: NL-SIX, Nextfeed, VoiceIX, NL-ix Live. Het bedrijf is gevestigd aan de Laan Copes van Cattenburgh 73, 2585 EW, Den Haag. (NLix website)

Het bedrijf word geleid door Jan Hoogenboom (CEO) en Mischa van Rossum (CFO). Jan is bij dit project zowel de opdrachtgever als de begeleider van dit project.

De organisatie bestaat op het startmoment van het project 12 mensen, maar zal in de loop van het project groeien. Het bedrijf bestaat uit grofweg vier afdelingen, maar vrijwel iedereen heeft binnen de organisatie een eigen rol. De afdelingen zijn: management, mid-office, sales en techniek. De opdracht zal worden uitgevoerd in opdracht van het management, maar om praktische redenen heeft de uitvoering op de afdeling techniek plaats gevonden.

Vlak voor het project van start ging is het bedrijf over genomen door KPN Wholesale, echter dit heeft geen directe invloed gehad op het project. De besluitnemers in dit geval zijn nog steeds de voormalig eigenaren van bedrijf. Het bedrijf heeft bij de overname ervoor gekozen niet te integreren met de KPN kantooromgeving en cultuur.

Hieronder staat een organigram dat duidelijk maakt wat de positie van het project is binnen het bedrijf. Het project staat direct onder leiding van het management en gebruikt resources van de overige afdelingen, maar is geenszins leidinggevend.



2-1 Organigram NLix

[1] Internet Exchange is een (commerciële) organisatie die autonome netwerken met elkaar via een switched netwerk verbind.

[2] VLAN, is een Virtueel LAN over het geswitchte backbone network. Dit product kan worden gezien als een

2.1.1 Management

Bestaande uit de CEO, CFO en CCO Marc Gauw houd het bedrijf op gang. Ze zetten de koers van het bedrijf uit en communiceren dit naar de KPN eigenaren van het bedrijf. Dankzij de grote groei van het bedrijf in de afgelopen jaren besteden vooral Marc en Jan veel tijd onderweg van en naar conferenties, bijeenkomsten met klanten en onderhoud van KPN relaties. Deze afdeling geeft de opdracht tot het uit te voeren project en levert Jan als bedrijfsmentor.

2.1.2 Mid-office

Deze afdeling is een administratieve taak toegeschreven, doet inkoop van het bedrijf en fungeert tevens als orderverwerking om de druk op de techniek en sales afdeling te verlichten. Momenteel is deze ingericht om eventuele fouten tussen het sales traject en de verwerking hiervan bij techniek af te vangen. Met deze afdeling heeft het project niet direct te maken.

2.1.3 Sales

Verzorgt uiteraard de verkoop van de producten en diensten van NLix. Deze mensen zijn de eerste lijn als het gaat om de invoer van orders, en de opvolging van nieuwe verkoopkansen. Met sales is er tijdens het project wel gepraat om de procesvoering binnen het bedrijf in kaart te brengen, maar speelt verder geen rol tijdens het project.

2.1.4 Techniek

Bestaat uit 6 man en is opgedeeld in twee teams. Eén team zorgt voor het onderhoud en de uitbreiding van de backbone van het bedrijf zelf. Deze mensen hebben veel te maken met backbone switches en interne BGP configuraties. Tevens onderhouden ze de contacten met de datacentra en fibercircuit leveranciers.

Het tweede team levert de klant producten en diensten. Hier word ook nog onderscheid gemaakt tussen de standaard producten en maatproducten die door verschillende werknemers geregeld worden. Hier moet bij opgemerkt worden dat alle mensen van Techniek genoeg weten van elkaars werk dat zij bij afwezigheid van de collega het werk kunnen overnemen.

Omdat de toekomstige afnemer van het project zowel techniek als management zal zijn, en ik hier op de afdeling gezeten heb, heb ik voornamelijk met deze afdeling te maken gehad.

2.2 Diensten en producten

NLix bied onder de verschillende merknamen een breed scala van producten aan. Grofweg zijn deze onder te verdelen in de Peering-, Carrier-, Community producten en Fiber diensten die worden afgeleverd. Daar het project de backbone van het bedrijf beslaat zal dit een korte beschrijving zijn. Uiteraard zijn de verschillende producten met elkaar te combineren. Hieronder word globaal uitgewijd wat de producten inhouden. Voor gedetailleerde informatie en een overzicht van alle producten verwijs ik naar de website van NLix.

2.2.1 Peering producten

Peering producten beslaan een breed spectrum en bevatten meestal een poort aan de verschillende switches die op 43 verschillende datacenter locaties staan (NLix statistieken). Aan zo'n aansluiting kan een aantal interne producten worden gekoppeld.

- Een privé VLAN[2]
- Een aansluiting op het VLAN van de NLix peering group
- Een aansluiting op het VLAN van de Joint Transit peering group
- Route server[3].

Vrijwel alle peering producten vereisen dat de klant via een NLix poort is aangesloten op het netwerk.

2.2.2 Carrier producten

Beslaat voornamelijk point to point of point to multipoint verbindingen die voor de klant worden opgezet over het NLix netwerk. Klanten kunnen hiermee met eigen netwerken met elkaar verbinden. Dit gebeurt tijdens het project nog via VLANs, maar zal later ook via MPLS worden aangeboden. Deze diensten worden geleverd over het bestaande NLix netwerk.

2.2.3 Community producten

Er worden over het backbone netwerk, naast IP verkeer ook specifieke oplossingen geboden voor Usenet[4], Voice over IP verkeer en multicast verkeer via NLix Live!.

2.2.4 Fiber diensten

Voor klanten met speciale wensen, of het uitbreiding van het netwerk tussen de bestaande locaties word er ook wel fiberglas verbindingen aangeboden. Deze worden meestal ingekocht bij één van de leveranciers of via de eigenaar KPN Wholesale. Het is tevens mogelijk over het netwerk via verschillende golflengtes (licht) een enkele vezel op te delen in 16 verschillende kanalen. Dit vergroot aanzienlijk de capaciteit en daarmee de mogelijkheid tot het verhuren van de lijnen.

2.2.5 BGP en IPv6 training

Als aanvullende dienst naar de klant bied NLix ook BGP cursussen en IPv6 training aan, aan klanten en geïnteresseerden. Deze trainingen worden gegeven door Ijitsch van Beijnum, auteur van

---

[2] VLAN, is een Virtueel LAN over het geswitchte backbone network. Dit product kan worden gezien als een leased line, daar deze op laag 3 transparant is.

[3] Een route server staat alle deelnemende klanten toe om automatisch alle BGP prefixes met elkaar uit te wisselen, zonder dat het adres van de routeserver naar buiten toe bekend word gemaakt.

[4] Usenet is een wereldwijd, gedecentraliseerd netwerk voor de uitwisseling van door de gebruikers gemaakte tekstberichten of binaire bestanden (Usenet op Wikipedia)

verschillende boeken, waaronder "BGP" uitgegeven door O'Reilly. (van Beijnum) Tijdens de beginfase van het project heb ik dankbaar gebruik mogen maken van de gegeven BGP en IPv6 training.

2.3 Beschrijving van het netwerk

Het NLix backbone netwerk bestaat uit een packet switched systeem. Er word onderscheid gemaakt tussen de core en peer sites. Op de core sites is in ieder geval alle hardware en zijn alle uplinks redundant uitgevoerd. Op de kleinere sites kan dit ook enkelvoudig zijn. Het onderscheid word voornamelijk gemaakt door de aangeboden switch capaciteit.

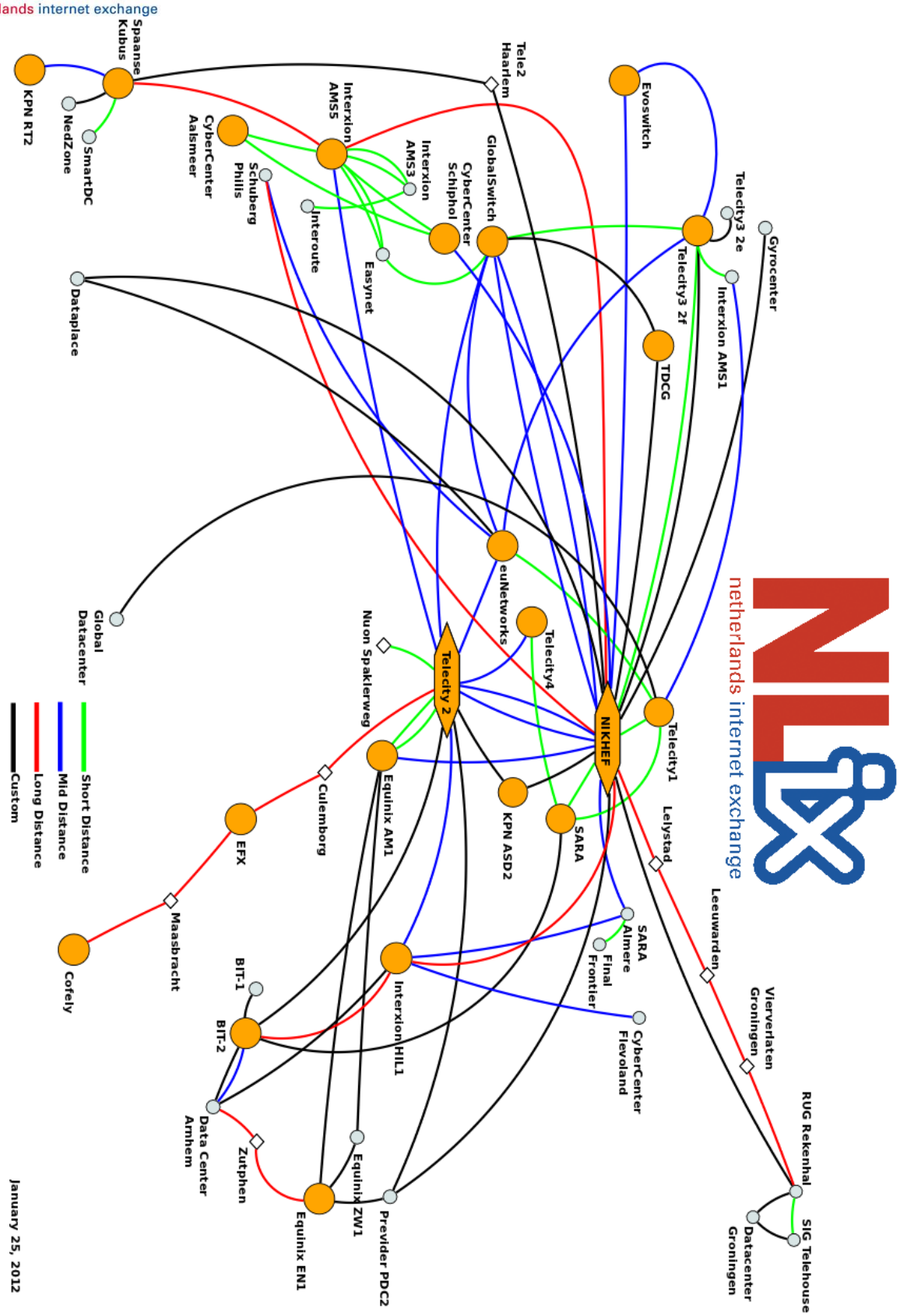
Op de zogenaamde core sites staan veelal Brocade MLX 16 of MLX32 switches, die respectievelijk 14 of 30 switch fabrics[5] accepteren. Deze switches accepteren 100mbit, 1gigabit of zelfs 10gigabit per poort aan capaciteit beschikbaar. Op de overige sites word gebruik gemaakt van kleinere Brocade MLX, XMRs of in sommige gevallen de oudere Foundry[6] Big- of Fast Iron switches.

Voor de fiber verbindingen word voornamelijk gebruik gemaakt van MRV optische multiplexers en media converters om bijvoorbeeld een glasvezel ethernet signaal over te zetten op een koper medium.

Op de volgende pagina staat een schematisch overzicht van het NLix netwerk met daarop de verschillende locaties en type verbindingen, zoals gepubliceerd op 25 januari 2012.

[5] Switch fabrics zijn insteek kaarten voor switch frames. Op een fabric, die als individuele switch functioneert, zit een eigen backplane en kan op verschillende snelheden intern of met de frame backplane pakketten switchen.

[6] Foundry Networks is per juli 2008 onderdeel geworden van Brocade Communication Systems.



2-2 NLix backbone netwerk op 25 januari 2012

2.4 Beschrijving van het netwerkbeheer

Het netwerk word door de afdeling techniek beheerd en dit gebeurt voornamelijk handmatig. De backbone is ontworpen, gebouwd en ingericht door de engineer Arno Griffioen en word mede beheerd door Lucas van Schouwen. Zij gebruiken voornamelijk terminal sessies om te verbinden met de switches en administratie van de systemen vind plaats in het door Jan Hoogenboom ontworpen database / intranet systeem.

Voor de monitoring word gebruikt gemaakt van een serie scripts die door de engineers zelf word onderhouden, het visualisatie systeem Cacti[7] en het netwerk polling en notificatie systeem Nagios[8].

Voor notificaties vanuit de switch en multiplexer systemen word er gebruik gemaakt van Syslog[9] functionaliteit. Deze meldingen worden op het intranet via een tool samengevoegd en worden dagelijks door de engineers doorgespit op zoek naar eigenaardigheden. Tevens word deze tool gebruikt om in het geval van storing terug te kunnen kijken naar de evenementen die daartoe geleid hebben.

Een toevoeging aan de toolset is een eigen ontworpen systeem genaamd NAL (Network Abstraction Layer) dat door engineer Vincent Bourgonjen is geschreven. NAL verzorgt binnen het terminal systeem, via telnet, automatisch de login gegevens voor de sessies met de switches, en handelt vooraf gedefinieerde acties aan om het netwerkbeheer te stroomlijnen.

2.5 Beschrijving van de situatie bij aanvang

Bij aanvang van het project is het bedrijf, na de overname, zojuist verhuisd naar een nieuw pand en ben ik niet de enige die zich moet in werken in een nieuwe situatie. Jan Hoogenboom (verder opdrachtgever) heeft aangegeven dat het bedrijf erg snel gegroeid is en dat om efficiënte besluiten te maken er meer informatie gebruikt kan worden. Naast informatie is er behoefte aan een extra controle laag.

Er word aangegeven dat in het proces van het accepteren van een offerte tot het uiteindelijk afsluiten van de klant er nog wel eens een foutje ontstaat tussen de administratie en het feitelijke netwerk. Zo komt het voor dat er voor een klant een netwerkpoort word gereserveerd terwijl deze nog in gebruik is. Dit word intern uiteindelijk opgelost, maar dit kan een lange levertijd opleveren als dit betekend dat er nieuwe hardware besteld moet worden in het geval er geen andere capaciteit beschikbaar is.

Aan de andere kant is er het probleem dat klanten die de dienst opzeggen niet altijd tijdig worden afgesloten. In het ergste geval kan dit een lange tijd zonder melding misbruikt worden.

Hoe het feitelijke bedrijfsproces werkt is bij aanvang onduidelijk en moet in kaart gebracht worden.

---

[7] Cacti: http://www.cacti.net/index.php
[8] Nagios: http://www.nagios.org/
[9] Syslog is een standaardisatie voor computer data logging. Het is een industrie standaard voor UNIX-achtige omgevingen en netwerk systemen.

2.6 Beeld van de opdrachtgever (IST/SOLL)

IST SOLL is een rode draad dat door het gehele project loopt. Het word door de opdrachtgever omschreven als een denkwijze, en word intern het bedrijf toegepast als methodiek.

IST is Duits voor het 'is' of 'zijn'. SOLL is Duits voor de 'het zal zijn'. Binnen NLix word IST SOLL opgevat als een controle model van de werkelijkheid. Het voorbeeld wat de opdrachtgever zelf noemt is facturatie. Indien een klant een dienst afneemt, zal deze daar ook voor moeten betalen. De SOLL is in dit geval: er moet een factuur verstuurd worden naar de klant. De IST geld hier als een controle mechanisme om te controleren of deze conditie ook waar is. De vraag bij IST geld dan: 'Er zou nu een factuur verstuurd moeten zijn naar de klant, is dit wel zo?'

De opdrachtgever heeft bij de oriëntatie aangegeven dat het eindproduct ter ondersteuning gaat dienen als tool om deze denkwijze te ondersteunen. De invulling hier aan ziet de opdrachtgever als volgt:

- **IST -** Een omstandigheid in het netwerk, bijvoorbeeld de status van een netwerkpoort.
- **SOLL -** De gewenste status van deze netwerkpoort als model van de werkelijkheid, zoals opgeslagen in de intranet database.

Waarbij er zodra er een onbalans is, dus de IST komt niet overeen met de gewenste SOLL situatie, moet er een notificatie uitgaan.

Hierbij moet opgemerkt worden dat notificaties niet leidend zijn, maar een indicatie naar zowel de afdeling techniek, als het management is. De afdeling techniek beheert zijn eigen prioriteiten en zal daarom in de praktijk het laatste woord hebben over de afhandeling van de notificaties.

Voor de implementatie heeft de opdrachtgever enige ideeën, die later in het proces worden uitgediept. Het generieke idee is dat er een webinterface komt die gekoppeld moet worden aan een email systeem. Tevens zullen er voor het communiceren met het netwerk of de database verschillende modules worden gekozen, geschreven of gebruikt worden.

Opgemerkt is dat er in deze oplossingrichting ook andere mogelijkheden zijn, zoals bijvoorbeeld het implementeren van Nagios, maar de opdrachtgever geeft aan dat dit niet voldoende integratie oplevert met het intranet en dat Nagios reeds voor monitoring word ingezet.

2.7 Probleemstelling

Het probleem dat zich voordoet is dat er tussen de afdelingen inconsistente klant en systeem data gevonden word in de vorm van klanten die niet afgesloten blijken en netwerkpoorten die reeds bezet zijn. Ondanks dat dit geen urgente problemen oplevert, immers de engineers of administratief personeel vangen dit van elkaar op, is het een potentieel financieel risico wat gedetecteerd en vermeden zou moeten worden.

2.8 Doelstelling

Binnen de gezette termijn van 17 weken moet inconsistentie in het IST SOLL model aan te tonen zijn door middel van een software tool dat rekening houdt met informatie uit de MySQL database en de live omstandigheden in het actieve backbone netwerk.

3 Invulling van het beheersproces

3.1 Cursus en kennismaking

Mijn eerste indruk is dat het bedrijf klein is opgezet, een informele werksfeer heeft en hard gegroeid is. Dat laatste is allereerst te merken aan de verhuizing naar een groter onderkomen, maar ook aan de werkdruk op mijn directe collega's.

Nog voordat het project van start ging greep ik de kans om mij bekend te maken met de materie door het volgen van de BGP cursus die door het bedrijf aan klanten gegeven word. Deze cursus word gegeven door Ijitsch van Beijnum en omvat de volgende onderwerpen:

- Routing
- IPv4 en IPv6 adressering
- Route selectie
- Route maps
- Peering groups
- Beveiliging
- 32bit AS nummers
- Load balancing
- Filtering
- Prepending

Aan het einde van de cursus heb ik samen met een klant een volledige BGP omgeving opgetuigd, waarna er succesvol BGP informatie is uitgedeeld tussen ons en de overige deelnemers. Ik merk tijdens de cursus wel dat mijn router terminal kennis wat roestig is, maar aan het einde van de cursus is dat geen probleem meer. Mijn voorgaande Cisco kennis lijkt toereikend.

Na de BGP cursus ben ik op eigen initiatief met collega Gert-Jan Aalderink mee gegaan voor een ronde routine onderhoud in Haarlem, EvoSwitch datacenter, Amsterdam, Equinix 2 datacenter en TeleCity TC2 datacenter waar Gert-Jan een 'wave' voor een klantorder ging aanleggen. Daar kreeg ik tevens de kans om te kijken welke apparatuur er door NLix gebruikt word en hoe zij de verbindingen regelen. Ik leer dat de werkwijze als volgt is:

- NLix plaatst een eindnode van het eigen netwerk in een datacenter, op verzoek van het datacenter of op verzoek van een klant. Dit is meestal in de vorm van een edge router met switch capaciteit, maar soms ook direct op een layer 3 switch, indien het om een grotere eindnode van het netwerk gaat.
- Deze eindnodes worden met elkaar verbonden met veelal redundante verbindingen naar andere locaties waar NLix apparatuur heeft staan. Dit gebeurt vrijwel altijd via een glasvezel verbinding, waarbij de verbinding tussen de datacentra door verschillende glasvezel leveranciers geleverd word. De verbinding tussen de NLix switch en de glasvezel leverancier word door het datacenter geleverd in de vorm van een patch.
- Een klant die op het NLix platform wil aansluiten moet zelf voor een (patch) verbinding naar het rack van NLix zorgen, alwaar NLix gebruik maakt van het patchpaneel van het datacenter. NLix patcht vervolgens zelf de verbinding door naar de poort of multiplexer waarop de klant word aangesloten.

Ik leer die dag dat het mogelijk is om via een enkele glasvezel aders door gebruik te maken van multiplexers er tot op 32 verschillende signalen over een enkele vezel kunnen. Dit staat NLix toe om een enkele gehuurde glasvezel verbinding zowel zelf te benutten, als ze toestaat om een transparante verbinding voor de klant te creëren.

Tevens kom ik achter het bestaan van mediaconverters, die data één op één doorgeven van bijvoorbeeld een glasvezel verbinding naar een koper verbinding. Dit levert echter de nare eigenschap op dat de verbinding moeilijk de debuggen is en dat de portstatus (up of down) niet altijd accuraat is. Dit kan gevolgen hebben in een later stadium, als blijkt dat er een mediaconverter achter een interne- of klant netwerkpoort zit.

Kort na de datacenter bezoeken is er nog de IPv6 cursus van de bovengenoemde Iljitsch, die de volgende onderwerpen behandeld:

- De technische werking van IPv6
- Het opraken van IPv4 adressen
- IPv6 diensten als DHCPv6 en DNS
- IPv6 beveiliging
- Tunnels
- Routing met IPv6 onder OSPF

Mijn collega's vertellen mij dat IPv6 waarschijnlijk weinig uit gaat maken, aangezien IPv6 binnen NLix nog niet helemaal geïmplementeerd is. Maar op dit punt kan extra kennis inslaan absoluut geen kwaad.

Over het geheel gezien ben ik anderhalve week bezig geweest met oriëntatie, meekijken met collega's en met het doorwerken van het cursusmateriaal.

3.2 Projectmethodiek

Nadat ik was voorzien van een tijdelijke werkplek, in afwachting van een normale werkplek in de ruimte bij andere collega's, ben ik aan de slag gegaan met het opzetten van het project.

De eerst genomen stap in het opzetten van het beheersproces was het overleg met de opdrachtgever dat kort na aanvang van het project plaats vond. In dit overleg word de communicatie met de opdrachtgever vastgelegd, namelijk:

" Begeleiding word op een vast moment ingepland. Vanwege de dynamische agenda van de begeleider (Jan zelf) zal hier flexibel mee worden omgegaan. Er word duidelijk gemaakt dat dit geen enorme druk op de agenda hoeft te zijn." - Gespreksverslag 31-05-11

Verder word er bepaald dat van de formele gesprekken een gespreksverslag word opgesteld, en deze zal door de opdrachtgever worden gevalideerd op de inhoud. Op het moment van het gesprek was er nog geen werkstation of werkplek voor mij beschikbaar, dus ben ik op een eigen omgeving aan de eerste versie van het plan van aanpak gaan werken.

De eerste stap in het proces dat de totstandkoming van het project was de keuze van projectmethodiek. In het oorspronkelijke plan, dat voor de aanvang van het project is opgesteld is sprake van een los onderzoek en een losse implementatie. Al snel realiseerde ik mij dat er in dit project geen sprake is van twee verschillende onderdelen, en dat een projectmethode gekozen moet worden die zowel een stuk onderzoek, als een stuk ontwerpen en implementatie controleert als één geheel.

Bijkomend bij het kiezen van een projectmethodiek is het vaststellen van de projectomgeving.

- Er is geen keuze voor projectmethodiek door de opdrachtgever voorgeschreven.
- Er is geen projectmethodiek die binnen het bedrijf actief word uitgevoerd.
- Er zijn geen collega's die kennis of ondersteuning kunnen leveren bij het uitvoeren van een specifieke methodiek.

Bij deze omgeving moet er bij het kiezen rekening gehouden worden met de volgende criteria.

1. *Het project word solo (niet in teamverband) uitgevoerd.*
2. *Het project moet in een relatief korte tijd tot een werkend eindresultaat komen.*

Specifieke criteria vanuit het oogpunt van de opdrachtgever is dat het project meetbaar is, en dus zal het met de gekozen projectmethodiek mogelijk moeten zijn om meet- en sturingsmomenten te creëren. Om die reden moet er ook een duidelijke fasering in het project zitten.

3. *Het project moet meet- en stuurbaar zijn voor de opdrachtgever.*
4. *Het project moet duidelijk gefaseerd zijn.*

Voor mijzelf heb ik aan de lijst met criteria toegevoegd dat de methode redelijk bekend moet zijn en dat de implementatie hiervan geen onnodige overhead oplevert, gezien de relatief korte duur van het project. Dit ook gezien er maar één persoon aan het project werkt. Overhead word hier ook gezien als het leren van een nieuwe stroming van projectmethodiek.

5. *Het project moet simpel te beheren zijn en geen overhead opleveren.*

Na aanleiding van deze criteria ben ik op zoek gegaan in mijn huidige achtergrond naar projectmethodes die ik in het verleden heb toegepast:

- **ASI**, een methodiek dat vooral word toegepast op netwerkarchitectuur. Het is lastig hier actuele documentatie en implementatievoorbeelden over te vinden.
- **RUP**, Rational Unified Process, een vrij breed gedocumenteerde projectmethodiek dat sterkt leunt op iteratie en integratie van verschillende ICT disciplines. Word vaak gecombineerd met UML als modellering.
- **Waterval** methodiek, een oncontroleerbare waterval waar de stappen elkaar opvolgen, ongeacht of deze succesvol zijn.

Vervolgens heb ik de lijst aangevuld met gerelateerde projectmethodes.

- **EUP**, de enterprise versie van RUP, kenmerkend door de toevoeging van 7 enterprise disciplines. Tevens omvat dit ook de cycli: Production & Retirement om de gehele levensduur van software te omvatten in de projectmethodiek.
- **Agile**, een collectie van best practises gevormd om iteratie op elke schaal binnen de projectmethodiek toe te kunnen passen. Is geschikt om overhead op het project te minimaliseren. Belangrijk is dat er geen duidelijke set van requirements hoeft te zijn voordat er aan de implementatie begonnen word omdat het elke cycle bij te sturen.

Hieronder is de keuze schematisch uitgewerkt:

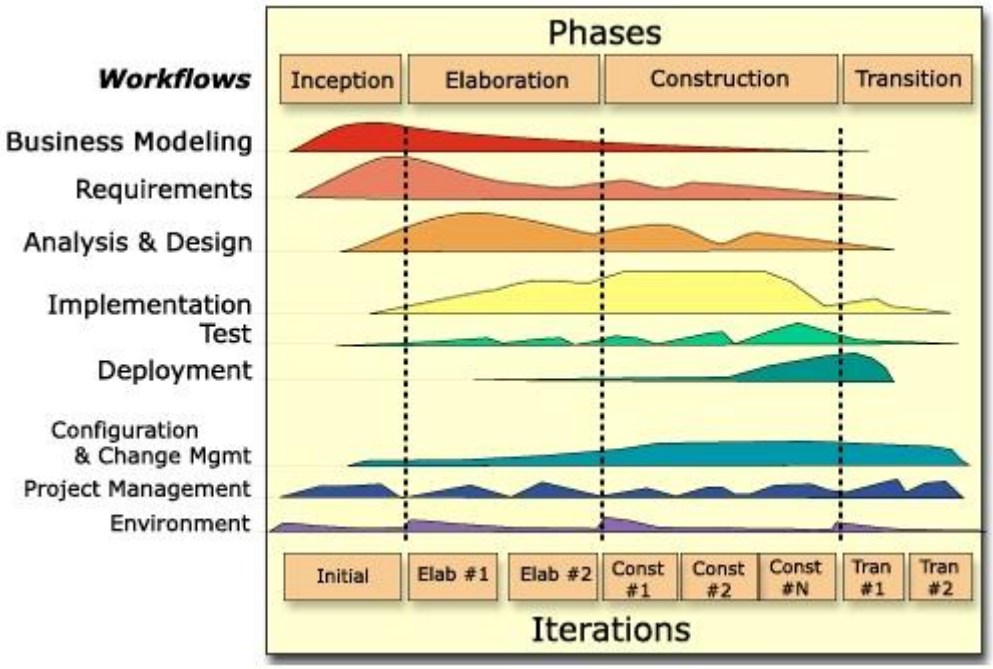
	Solo	Resultaat	Stuurbaar	Fasering	Overhead	**Totaal**
ASI	+	+	+	+	+	**+**
RUP	+	++	+	++	++	**++**
Waterval	+	++	--	--	++	**+/-**
EUP	--	++	+	++	--	**+/-**
Agile	+	++	++	--	+	**+**

Tabel 1 Keuze projectmethodiek

De keuze valt op **RUP**. ASI is minder geschikt vanwege de summiere documentatie en het feit dat het meer geschikt is om netwerkarchitecturen te ontwerpen. De watervalmethodiek kent vrijwel geen fasering, EUP levert teveel overhead op en Agile kent geen vaste fasering, wat het plannen erg lastig maakt.

3.3 RUP

Rational Unified Process (verder RUP) is in 2003 ontwikkeld door IBM als een flexibele projectmethodiek die iteratieve software ontwikkeling ondersteund. (Ambler S. W., ManagersIntroToRUP.pdf, 2005)



3-1 Fasering en modellering RUP (RUP Fundamentals)

Elke fase sluit af met een milestone review. Dit is een evaluatie met de opdrachtgever, om te controleren of het project aan de vooraf gestelde eisen voldoet en voortgang boekt. Dit is een belangrijk onderdeel van de kwaliteitscontrole binnen RUP. Tevens levert elke fase een eindproduct op, wat bij de milestone besprekingen beoordeeld word op inhoud en kwaliteit.

3.3.1 Fasering

RUP kent de volgende fasering:

Inception fase. In deze fase is het doel om alle eisen en wensen boven tafel te krijgen. Er kan een begin gemaakt worden met de modellering door voor een ontwerprichting te kiezen. Aan het einde van deze fase moet er vast liggen dat:

- De scope van het project vast ligt.
- De initiële eisen en wensen boven tafel zijn, voor zover dat mogelijk is.
- De risico's zijn geïdentificeerd en vastgelegd.
- Het duidelijk is voor de opdrachtgever dat er resultaat geboekt kan worden.
- Dat het beheerproces degelijk ingericht is.

Elaboration fase. In deze fase worden de specificaties verder uitgewerkt en zal de architectuur ontworpen worden. Het moet in deze fase duidelijk worden of het project haalbaar en de risico's en randvoorwaarden moeten hierbij benoemd worden. Aan het einde van deze fase ligt het volgende vast:

- Het project is realistisch en haalbaar.
- De eisen en wensen zijn tot dan toe vastgelegd.
- De architectuur en het ontwerp zijn goedgekeurd.
- De risico's zijn zover aanwezig afgedekt.
- De detailplanning is uitgewerkt voor in ieder geval de Construction fase.

Construction fase. In deze fase word het systeem ontwerp uitgewerkt en daadwerkelijk uitgevoerd. Hierbij ligt de nadruk op praktische oplossingen, gebruikerstesten en het meten van de resultaten met de gestelde eisen en wensen. Aan het einde van deze fase moet het volgende vast liggen:

- Het software ontwerp en de ondersteunende documentatie zijn van een acceptabel niveau.
- De opdrachtgever is bereidwillig het ontworpen systeem in gebruik te nemen.
- De risico's zijn gedurende het gehele project beheerst.
- De verdere planning staat vast.

Transition fase. In deze fase word het ontwerp getest, gevalideerd en opgeleverd. Dit omvat zowel de implementatie, als de testrapporten en handleidingen. Deze fase levert het volgende op:

- Het systeem, inclusief documentatie.
- Eventuele gebruikershandleidingen.
- Het systeem is operationeel.
- Het systeem kan effectief ondersteund en ingezet worden.

3.3.2 Disciplines

Naast een fasering RUP, een negental disciplines. Zes van deze disciplines worden ingezet om het project per fasering te beheren. De laatste drie (*schuin afgedrukt*) zijn continue actief en zijn een handleiding om het project in z'n geheel te monitoren. De RUP disciplines staan hieronder vermeld met een korte beschrijving van de acties per discipline.

- **Business Modeling**. Het inzichtelijk maken van de (aanvang)situatie binnen het bedrijf, het systematisch onderzoeken en documenteren van de bedrijfsprocessen. Het identificeren van de verantwoordelijkheden en rollen.
- **Requirements**. Het proces waarin de eisen en wensen in kaart gebracht worden. Het opzetten van de middelen hiertoe (zoals een interview met de opdrachtgever), het vastleggen van de resultaten en de communicatie tussen de opdrachtgever en het project.
- **Analysis and Design**. Het formuleren van een concept architectuur. Het construeren van een proof-of-concept ter validatie van het ontwerp. Het meten van de systeemeisen met het proof-of-concept. Het vastleggen van het ontwerp, de gebruikers interfaces en het database ontwerp.
- **Implementation**. Het transformeren van het ontwerp in uitvoerbare code. Inclusief het testen, waaronder ook het gebruik van unit-testing. Ook het integreren met beschikbare bibliotheken, en het testen hiervan.

- **Testing**. Het vastleggen van de testeisen, het testplan. De uitvoering van het testplan en het rapporteren van de testresultaten.
- **Deployment**. Het plannen, organiseren en uitvoeren van de uitrol van het systeem.
- ***Configuration Management***. Het bijhouden van projectdocumentatie. In deze discipline is het doel om alle wijzigingen door te voeren, en het toepassen van bijvoorbeeld versienummering.
- ***Project Management***. Het beheren van de risico's, wijzigingen, beschikbare mensen en de communicatie met de opdrachtgever.
- ***Environment***. Deze discipline heeft als doel om alle benodigde middelen tijdig in te richten voor gebruik. Dit gaat met name over werkstations, software en later in de Transition fase, de benodigde hardware en software voor de implementatie.

De intentie is om RUP direct, zonder verdere mutaties te implementeren. In de planning zal rekening worden gehouden met 2 iteratie cycle in de constructiefase.

4 Inception fase

In navolging van de keuze voor RUP is de tweede stap het inrichten van het beheersproces. Deze fase word gekenmerkt door de volgende processen:

- Het bepalen van de scope van het project.
- De beschikbare resources worden ingericht en toegekend.
- De risico's vastleggen.
- De haalbaarheid van het project word bepaald.
- Het vergaren van aanvullende, benodigde informatie.
- Het invulling geven aan de planning.
- Er word een begin gemaakt aan de oplossing.

Nu is technisch gezien voor de keuze van het beheersproces al kennis over het project nodig, en dit is in de eerste twee weken van de stage vergaard.

De invulling van de disciplines ziet er in AUP als volgt uit:

- **Model**
  - Het vastleggen van de requirements in een globaal model.
  - Het vastleggen van de architectuur in een globaal model.
- **Implementatie**
  - Optioneel, een technisch prototype.
  - Optioneel, een user interface prototype.
- **Test**
  - Het opstellen van de testeisen.
  - Het begin maken van het testplan.
- **Deployment**
  - De release data vastleggen in de planning.
  - Algemene planning voor de uitrol.
- **Configuration Management**
  - Het opzetten van de werkomgeving.
- **Project management**
  - Het bepalen van de haalbaarheid van het project**.**
  - De globale planning opstellen.
  - De risico's beheren.
  - Het afsluiten van de fase dmv communicatie met de opdrachtgever.
- **Environment**
  - Het opzetten van de werkomgeving.

De Inception fase sluit af met het opleveren en de goedkeuring van een Vision document.

4.1 Eisen en wensen

De Inception fase begint voor dit project met een interview met de opdrachtgever en de eindgebruikers. Er is bewust gekozen beide partijen te interviewen om er zeker van te zijn dat er geen eisen of wensen zijn die de opdrachtgever mist, of dat het personeel vanaf hun positie eisen of wensen als overbodig ervaren.

Vooraf aan de interviews heb ik mijzelf de volgende vragen gesteld:

- Hoe kan ik de volledigheid van de antwoorden controleren?
- Hoe betrouwbaar zijn de resultaten van het interview?
- Hoe kan ik verifiëren of de gegeven antwoorden juist zijn?
- Hoe voorkom ik interpretatie fouten?

Het antwoord daarop was vrij simpel, *peer review*. Door mijn bevindingen te controleren tijdens het interview, door middel van doorvragen, opsommingen en concrete voorbeelden te stellen. En na het interview, door het maken van een gespreksverslag, en om deze te laten verifiëren door de opdrachtgever. Tevens is het via peer review mogelijk om collega's elkaar te laten corrigeren en aan te vullen. Dit is vooral bij Techniek toegepast, zeker omdat iedereen daar in één ruimte zit, en kan meepraten.

4.1.1 Interview met opdrachtgever

Het doel van het interview met de opdrachtgever is:

- Het vastleggen van de scope van het project.
- Het vastleggen van de duur van het project.
- Het vastleggen van het probleem.
- Het vastleggen van de doelstelling.
- De eisen aan de oplossing.
- De wensen aan de oplossing.
- De haalbaarheid van het project te bepalen.
- De risico's bepalen

Het interview word begin juni afgenomen, kort na de aanvang van het project. Het directe resultaat van dit interview is verwerkt in het plan van aanpak (ook wel Inception document).

Een korte samenvatting van de bevindingen:

- Te realiseren door mijzelf, waarbij ik ondersteund word door de engineers, het management en op verzoek de rest van het personeel.
- De duur van het project is 17 weken.
- Het probleem is dat inconsistente data tussen de database en het netwerk kan zorgen voor negatieve financiële problemen. Ondanks dat engineers zelf het werk controleren, is er behoefte aan een extra controle.
- Het doel is het creëren van een oplossing voor het aangegeven probleem, binnen de gestelde tijd van 17 weken, op basis van het IST SOLL model.
- Acceptatie is een werkende tool, waarbij er verder geen eisen gesteld worden aan de condities hiervan.
- Wenselijk is dat de tool gebruikt kan worden om extra informatie aan de engineers te leveren, zoals extra monitoring van netwerkpoorten.
- Het project moet zonder meer haalbaar zijn, voorwaarden zijn: een degelijke ontwikkelmethodiek, zorgvuldige begeleiding, toegang tot alle benodigde middelen en informatie. En als laatst moet er een mate van kwaliteitswaarborging gebruikt worden (zie hoofdstuk 6 kwaliteitswaarborging in het Vision document)
- De risico's zijn er volgens de opdrachtgever niet, omdat er niet aan bedrijfskritieke omgevingen word gewerkt.

Het interview word afgerond, en de notities worden door de opdrachtgever goedgekeurd.

4.1.2 Interview met sales

Het doel van het interview met sales, in dit geval Vincent van Gentevoort, is:

- Bepalen van het profiel van een gemiddelde klant.
- Het bepalen van mogelijke mutaties bij klanten, en de invloed hiervan op de infrastructuur.

Het interview, afgenomen op 27 juni heeft het volgende resultaat.

Er word vanuit Sales de belofte gedaan, na aanleiding van het interview, een klantprofiel uit te werken, maar in de praktijk komt dit profiel er nooit. Ook na contact met de opdrachtgever word deze belofte niet nageleefd.

De mutaties die plaats kunnen vinden zijn geïdentificeerd als:

- Aanname van order.
- Verhuizing van de klant, naar een ander datacenter.
- Naamswijziging van de klant.
- Overname van klant, door bestaande klant.
- Overname van klant, door externe partij, nog geen klant van NLix.
- Afsluiten van een order, na uitblijven van betaling.
- Afsluiten van een order, na beëindiging contract.

Deze mutaties hebben invloed op de volgende onderdelen van de infrastructuur:

- NLix poorten
- VLANs
- AS objecten (administratief)
- GSM modems voor remote beheer
- BGP router informatie, bij managed beheer en specifieke klantsituaties.

Het interview word samengevat in een kort gespreksverslag dat geverifieerd word door Vincent.

4.1.3 Interview Techniek

Bij de afdeling techniek gaat het voornamelijk om de inventarisatie van de werkomgeving. Immers de te ontwikkelen tool heeft voornamelijk de engineers als doelgroep. Graag wil ik weten waar techniek behoefte aan heeft, en of het product dat management voor ogen heeft ook daadwerkelijk toegepast gaat worden.

Bij deze interviews ben ik één voor één de engineers langsgegaan om ze de volgende vragen te stellen:

- Wat is de rol van de engineer binnen techniek?
- Welke systemen en software word er gebruikt voor het uitvoeren van de werkzaamheden?
- Heeft de engineer te maken gehad met het probleem dat de opdrachtgever stelt?
- Hoeveel waarde word er aan usability van de software gehecht?

Verder heb ik veel ruimte gelaten voor eigen inbreng, daar de engineers veel ervaring en een sterke mening lijken te te hebben over de opdracht.

De bevindingen kunnen als volgt worden samengevat:

- Techniek bestaat uit:
    - Vincent Bourgonjen - Service provisioning, verantwoordelijk voor het opleveren van standaard klantproducten.
    - Ger-Jan Aalderink - Customer projects, verantwoordelijk voor het opleveren en onderhoud van niet-standaard klantproducten.
    - Marco Kuiper - Customer support, verantwoordelijk voor binnenkomende support calls en tickets. Hij zal uiteindelijk de notificaties voor het NOC af vangen, en is daarom een directe eindgebruiker.
    - Arno Griffioen, Backbone Manager, verantwoordelijk voor de hardware kant van het backbone netwerk.
    - Lucas van Schouwen Routing Manager, verantwoordelijk voor de software kant van het backbone netwerk, en administratief contact voor routing zaken.
- Er word voor beheer gebruik gemaakt van:
    - Standaard SSH en Telnet clients
    - NAL (Network Abstraction Layer) een script geschreven door netwerk engineer Vincent. Dit script handeld de login sequentie voor de netwerkswitches af en bevat macro's voor veelgebruikte functies.
    - Syslog aggregators op het intranet
    - PHPmyAdmin, voor het database beheer. Maar ook voor de manuele invoer van gegevens.
    - Het intranet, voor het invoeren, bewerken en verwijderen van informatie.
- Vrijwel iedereen kent de problematiek gesteld door de opdrachtgever. Echter is iedereen het er over eens dat de mate waarin inconsistente data voor financiële gevolgen zorgt minimaal is. Er word gesteld dat het eigenlijk helemaal geen probleem is, maar dat het best wel bewezen mag worden.

- Er word tijdens de interviews aangegeven dat er nul waarde aan de interfaces gehecht word. Het intranet toont dit ook wel een beetje aan. Het intranet is, vanuit het oogpunt van usability, niet voorbij het jaar 2000 gekomen. Zolang alles maar werkt is het devies.

Aanvulling op het interview word gegeven:

- Er word gezegd dat voordat er software binnen het bedrijf gebruikt zal worden, het zich eerst moet bewijzen. Er word feitelijk gezegd dat er draagvlak gecreëerd moet worden. Het lijkt daarbij niet uit te maken welke testen er worden uitgevoerd. Bij het doorvragen levert het geen extra test condities op.
- Waar nodig is, zullen er zelf (door de engineers) tools worden geschreven word mij enkele malen vermeld. Dit doet mij afvragen of dit project wel nodig is. De bevestiging komt alleen maar uit de hoek van het bewijzen of de IST SOLL aanpak ook echt resultaat levert.
- Documentatie die beschikbaar is, op bijvoorbeeld het Intranet, zal per definitie oud zijn, en niet voldoende informatie kunnen geven. Dit is een bekend probleem, maar het loont niet om hier tijd aan te besteden volgens de engineers.
- Er is een test laboratorium aanwezig, wat ingezet word om configuraties op te bouwen en eventueel te testen. Echter, alle wijzigingen in bijvoorbeeld de database en de productie omgeving zijn real-time en kunnen niet zonder meer terug gerold worden. Het enige wat de database beschermt tegen evt. fouten is een backup van diezelfde vroege ochtend.

Het interview word afgerond met een rondvraag aan alle engineers om aan te wijzen, of aan te vullen welke waardes er in het netwerk gemonitord moeten worden om volledig en zeker te kunnen zeggen dat er geen financiële problemen kunnen ontstaan.

De conclusie is, dat er tijdens de Elaboration fase goed onderzocht moet worden welke items dit nu precies zijn, aangezien het niet direct bekend is welke waardes er überhaupt beschikbaar zijn.

4.2 Resources

Hieronder word vermeld welke resources nodig zijn tijdens de uitvoering van het project. Deze resources worden bij de eerste inventarisatie nodig geacht.

Tevens word de volgende software nodig geacht:

- Open Office, als bedrijfsstandaard voor document publicatie.
- IDE (Intregrated Development Environment) naar eigen keuze in te richten, er is geen standaard software pakket voor beschikbaar.
- PHPMyAdmin[10], voor het beheer van de MyISAM MySQL database.
- Sourcecode van het huidige Intranet, ter oriëntatie.

Qua hardware is er het volgende nodig:

- Een werkstation waarop de bovenstaande software te gebruiken is. In de praktijk word dit na de tweede week een PC werkplek met daarop het Linux Debian 6 besturingssysteem.
- Toegang tot netwerk devices, om informatie uit te lezen.

NLix geeft absoluut de voorkeur aan Open Source software, daar het geen licentiemodel heeft en de ondersteuning daarop voor eigen rekening word genomen. Dat leek me een uitdaging, daar niet alle software pakketten direct open source beschikbaar zijn. In de praktijk gaf dit geen noemenswaardige problemen omdat ik direct met collega's kon overleggen als ik op zoek was naar specifieke software pakketten.

Personeel

- Minimaal één contactmoment per week met de begeleider.
- Tijd bij directe collega's om uitleg te vragen.
- Ondersteuning bij Intranet software, indien nodig.

Tevens is er nog documentatie.

- Het handboek voor de engineers, te vinden op het Intranet.
- Toegang tot handleidingen van de gebruikte routers en switches. Deze zijn veelal alleen bereikbaar voor accounthouders bij de fabrikant.
- BGP documentatie, in de vorm van boeken en online materiaal.
- PHP documentatie, in de vorm van boeken en online materiaal.
- MySQL documentatie, in de vorm van boeken en online materiaal.

Alle benodigde resources zijn geleverd, minus de contactmomenten, deze worden in overleg met de begeleider alleen ingelast als dat nodig is. In de praktijk betekend dit, als de agenda van de begeleider dit toestaat.

[10] PHPMyAdmin is Open Source software dat op basis van PHP, het beheer van MySQL databases toestaat. Voor meer informatie: http://www.phpmyadmin.net

4.3 Kwaliteitscontrole

De kwaliteitscontrole bestaat uit een serie opgestelde eisen, zoals vastgelegd in het Vision document (hoofdstuk 6) Deze eisen zijn opgesteld naar het inzicht op dat moment, en zijn gebaseerd op RUP waardes en in overleg met de opdrachtgever. Elk van de mijlpalen en producten dient gevalideerd te worden door de opdrachtgever.

Ik besluit als aanvulling op de kwaliteitscontrole een testcase op te zetten. In overleg met de opdrachtgever, en na aanleiding van het interview met Sales word begonnen met de opzet van de testcase 'MySrv', een fictieve klant met een doorsnee klantprofiel. Omdat ik op dat moment geen inzicht heb in welke producten er veel verkocht worden en hoe het product portfolio in elkaar zit besluit ik in overleg dit aan Vincent over te laten, daar die aangeeft dat voor mij te kunnen doen. De invulling van de testcase word, omdat het druk is bij Sales en op dat moment nog bezig is met het inwerken van nieuw personeel, verplaatst naar de Elaboration fase.

Het argument om een testcase te gebruiken is dat er geen huidige testomgeving bestaat, en er absoluut niet getest mag worden met echte klanten. Dit is de enige oplossing om externe variabelen, en daarbij mutaties, te kunnen nabootsen.

4.4 Globale planning

De globale planning, word gebaseerd op de fasering zoals die in RUP gevonden kan worden. Hierbij is de Invision fase bewust kort, omdat het vermoeden bestaat dat er veel tijd nodig is voor de Elaboration fase en de Construction fase. Voor de Transition fase word aanzienlijk minder tijd uitgetrokken, daar dit naar alle schijn een enkelvoudige applicatie word, zonder multi-platform eisen, cluster omgevingen of een ander type applicatie dat een grootschalige uitrol vereist.

Tevens word er rekening gehouden met enige uitloop tijdens het project. Er zit een totaal van 2 weken slack ingebouwd, door aan zowel de Elaboration, als de Construction fase een extra week toe te voegen.

Het resultaat is opgenomen in het Visie document als het hoofdstuk Globale planning.

4.5 Log registratie

Als onderdeel van het projectbeheer kies ik ervoor gedetailleerde logboeken bijgehouden. Dit bied de mogelijkheid snel terug te zien waar keuzes zijn gemaakt, en op welke basis dit is gedaan.

De logboeken zijn een verzameling digitale en papieren notities die met enige regelmaat zullen worden verwerkt. Een voorbeeld van de uitwerking is te vinden in hoofdstuk 12.8. Door middel van kleurcodering is dit in een enkele opslag in te zien. Er word onderscheid gemaakt tussen:

- Proces voortgang (geel)
- Keuzemomenten (oranje)
- Aannames (groen)
- Overige, belangrijke, notities (blauw)

De resultaten van de inception fase worden samengevat in het plan van aanpak, dan ook wel vision document genoemd en naar de opdrachtgever verstuurd, in afwachting van goedkeuring.

5 RUP naar AUP

Na het inleveren van het inleveren van het Vision document, in afwachting van de goedkeuring besluit ik nog wat dieper in de RUP methodiek te duiken, daar ik het idee krijg dat mijn kennis tot dan toe misschien niet toereikend is.

5.1 Problemen met RUP

Ik kom er al snel achter dat ik het document verkeerd benoemd heb, in de RUP terminologie moet dit Inception document zijn, niet het wat algemenere Vision document, of vertaald: Visiedocument.

Tevens kom ik er achter dat RUP meer overhead heeft als verwacht. Als ik begin met de detailplanning kom ik er achter dat de elaboratie fase een groot gebied bestrijkt, namelijk er word verwacht dat het ontwerp al zo goed als feilloos beschikbaar is, inclusief test-cases en het testplan. En ik realiseer me dat dit binnen de elaboratiefase misschien niet mogelijk gaat zijn.

Ik vermoed dat er juist in het onderzoek wat er gedaan word in de Elaboration fase een hoop informatie naar boven zal komen dat de Construction fase kan beïnvloeden. Zeker omdat ik het idee heb dat de opdrachtgever, die van nature nogal impulsief is, nog enigszins van koers wil wijzigen zodra hij een idee heeft van de uitwerking.

Een ander probleem is dat de opdrachtgever graag rapid prototyping terugziet in de software. Hiermee bedoel ik dat er binnen het bedrijf er een cultuur hangt om software on-the-fly te ontwikkelen, a la waterval methode. Dat is nog steeds geen reden voor mij om ook zo te werken, immers weet ik uit ervaring dat een dergelijke methode gigantisch uit de hand kan lopen. Zeker met een project als dit waar een vast termijn aan zit is dit garantie om te mislukken.

Tevens is er nog geen test-case, vanwege de vertraging bij Sales. Dit tezamen maakt RUP niet meer persé de geschikte candidaat. Ik moet dus op zoek naar een alternatief, wat mogelijk wel geschikt is. Echter, dit mag zo goed als geen impact hebben op de voortgang van het project, anders komt het project al vanuit de startpositie tot stilstand.

Dit leid mij ertoe te besluiten een extra eis voor de projectmethodiek in te voegen:

6. *De methodiek moet rekening houden met de dynamiek van aangepaste eisen en wensen tijdens het project. Deze dynamiek is nodig voor prototyping.*

Via de RUP documentatie kom ik uit op de tot dan toe nog niet belichte AUP. Agile Unified Process.

AUP heeft alle voordelen van RUP, maar kent minder overhead. Het neemt ook het beste van Agile, namelijk de flexibiliteit om tot in de Construction fase nog modellering en de testomgeving aan te passen.

Als AUP toegevoegd word aan de tabel met de oorspronkelijke keuze gebeurt er dit:

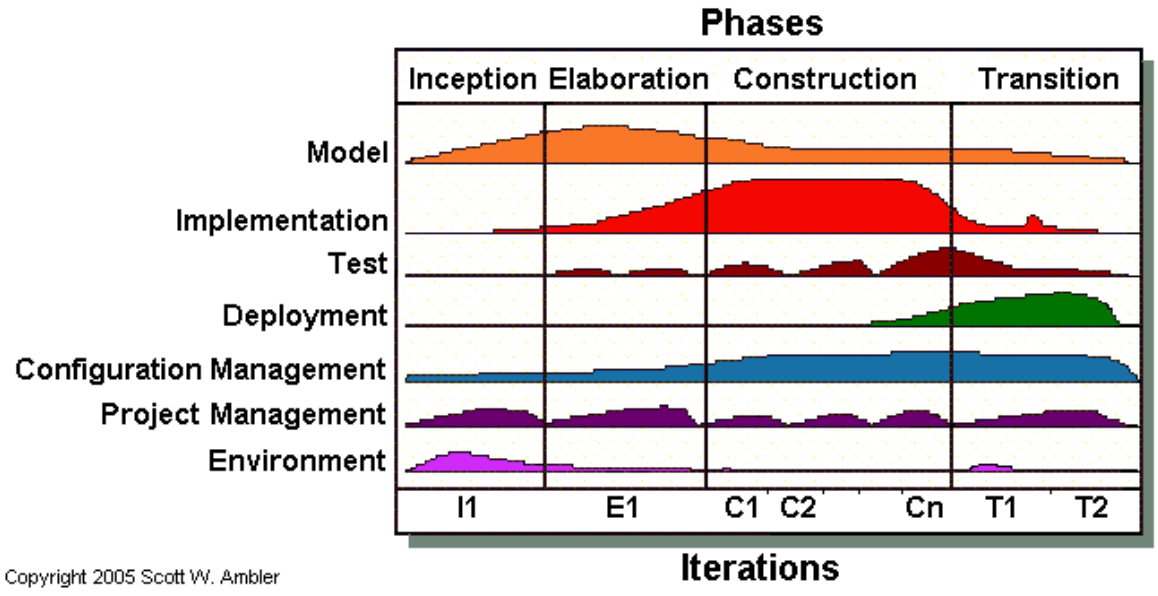
	Solo	Resultaat	Stuurbaar	Fasering	Overhead	Dynamiek	**Totaal**
ASI	+	+	+	+	+	+	**+**
RUP	+	++	+	++	++	-	**+**
Waterval	+	++	--	--	++	--	**-**
EUP	--	++	+	++	--	+	**+/-**
AUP	+	++	+	++	++	++	**++**
Agile	+	++	++	--	+	**+**	**+**

AUP is duidelijk een geschikte methodiek.

Na overleg met de opdrachtgever word besloten, om het proces niet te vertragen verder voort te zetten onder de AUP methodiek. Dit behoeft geen aparte rapportage in de vorm van een tweede visie document.

5.3 Agile Unified Process

Agile Unified Process (verder AUP) is bedacht door Scott Ambler (Ambler, 2009) als een alternatief op RUP. Hieronder staat AUP schematisch uitgewerkt.



5-1 AUP project cyclus (Ambler S. W., 2005)

AUP kent de volgende fasering:

- **Inception fase**. In deze fase word de scope van het project bepaald en word er gekozen voor een potentiële architectuur. Benodigde aanvullende informatie word in deze fase ingewonnen. Tevens vind er de verdeling van de benodigde middelen plaats en zal er na deze fase een goedkeuring plaatsvinden.
- **Elaboration fase**. In deze fase word de architectuur van het te ontwerpen systeem gemaakt.
- **Construction fase**. De implementatie vind in deze fase plaats, waarbij er gebouwd word op een Agile manier, dus een geprioriteerde lijst waarbij de belangrijkste features eerst worden gebouwd. In Agile worden deze lijsten dagelijks bijgewerkt, en per cycle geüpdate.
- **Transition fase**. In deze fase word het ontwerp getest, gevalideerd en opgeleverd. Dit omvat zowel de implementatie, als de testrapporten en handleidingen.

Naast een fasering kent AUP, net als RUP, een aantal disciplines:

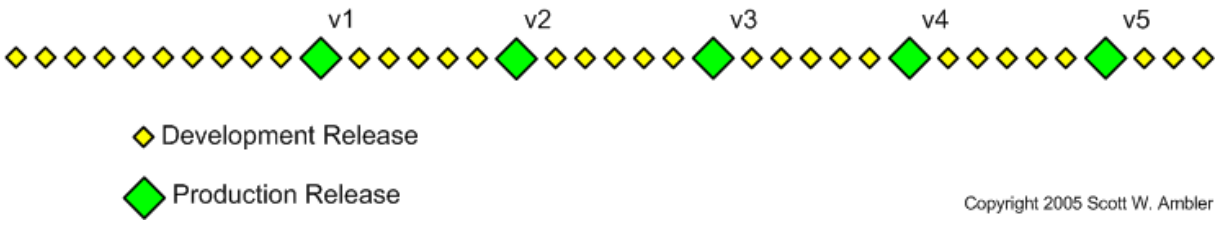
- **Model**. Modellering, hierbij is het doel de organisatie te doorgronden en dat proces te documenteren. Tevens is het de discipline waarin het probleemdomein bepaald word en er gewerkt word aan een oplossingsmodel.
- **Implementation**. De discipline waarin het ontwerp uitgevoerd word in uitvoerbare code. In deze discipline zit het unit-testen[11] inbegrepen.
- **Testing**. In deze discipline is het doel om de kwaliteit van het project en het product te waarborgen. Deze discipline is actief vanaf de Elaboration fase. Kwaliteitscontrole houd in dat er

[11] Unit-testen refereert naar het systematisch, op een zo laag mogelijk niveau binnen de software controle uitvoeren op de 'waarheid' van de uitvoering van functies. Deze resultaten worden getoetst met vooraf bepaalde verwachte resultaten.

gekeken word of de implementatie ook daadwerkelijk voldoet aan de gestelde eisen, en het proces van de validatie hiervan.

- **Deployment**. Het plannen en uitrollen van de implementatie met het doel deze beschikbaar te stellen voor de eindgebruiker. Dit zal voornamelijk in de Transition fase gebeuren.
- **Configuration Management**. Het bijhouden van projectdocumentatie. In deze discipline is het doel om alle wijzigingen door te voeren, en het toepassen van bijvoorbeeld versienummering.
- **Project Management**. Het beheren van de risico's, wijzigingen, beschikbare mensen en de communicatie met de opdrachtgever.
- **Environment**. Deze discipline heeft als doel om alle benodigde middelen tijdig in te richten voor gebruik. Dit gaat met name over werkstations, software en later in de Transition fase, de benodigde hardware en software voor de implementatie.

Naast deze disciplines en fases is er nog het Agile principe van de korte ontwikkel sprints. In overleg met de opdrachtgever is besloten deze, met oog op de korte tijd, alleen tijdens de Construction fase uit te voeren. Het doel is ten minste 2 van deze cycli uit te voeren, waarbij er eerst aan een ruw prototype gewerkt word, en in de 2e cycle dit eerste prototype verder uitgewerkt word met features als een webinterface en een configuratie tool.



5-2 Iteratie cycli (Ambler S. W., incrementalReleases, 2005)

Met het kiezen van de projectmethodiek begint het daadwerkelijke project met het uitvoeren van de Inception fase, waarin de inrichting van AUP voor dit project verder uitgewerkt word in de vorm van een set eisen en wensen, planning en een richtlijn voor het ontwerp.

5.4 Gevolgen voor het proces

De gevolgen voor het proces zijn minimaal. RUP en AUP hanteren vrijwel dezelfde methodiek in de Inception fase. Het verschil tussen de twee methodes zit hem in de manier van plannen, het algemene projectbeheer wat bij AUP iets effectiever is en het verschil word pas duidelijk in de elaboratie en implementatiefase. In de elaboratiefase is minder in omvang ten opzichte van RUP.

In de implementatiefase word er actief van prototypen gebruik gemaakt na de Agile methodiek. Waarbij korte ontwikkel sprints gebruikt worden om zo ruimte te geven voor veel sturing. Iets wat gewenst is bij dit project.

En bijkomstigheid is dat het risico op feature creep op deze manier verminderd word. Er word slechts per sprint aan een selecte lijst features gewerkt.

5.5 Detail planning

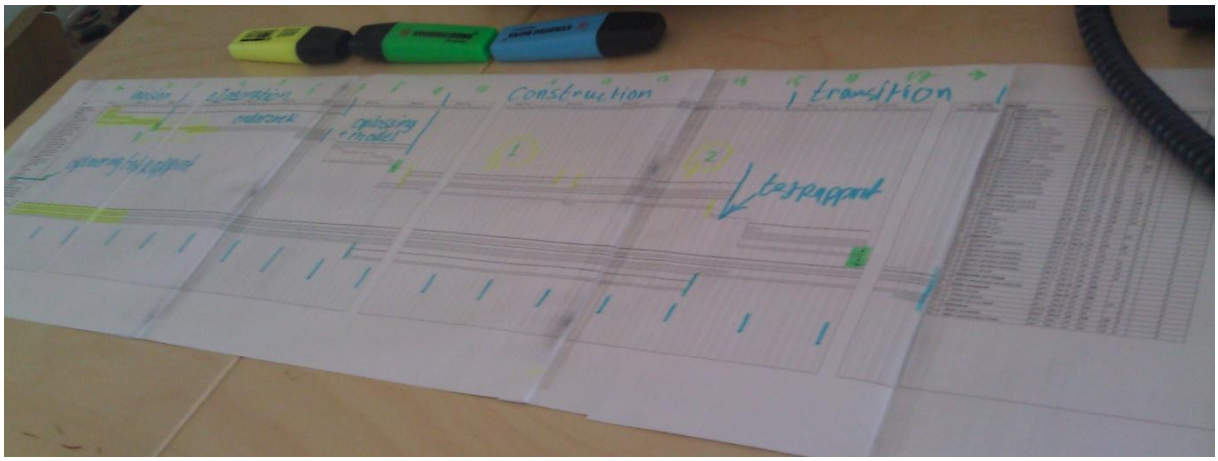
Om AUP te implementeren heb ik gekozen om de vooruitgang te monitoren en de planning in te richten door middel van een GANTT kaart. Gezien de directe link met de RUP familie en het feit dat GANTT mij veel voordeel bied op de planning heb ik besloten niet verder op zoek te gaan naar een alternatief.

De motivatie voor de keuze van een GANTT kaart kan opgesomd worden als:

- GANTT planning is een industriële standaard. Omdat deze methode van planning veel toegepast word zijn er veel (Open Source) software pakketten op de markt die het snel en efficiënt inrichten van de planning mogelijk maken.
- Het is een communicatiemiddel naar mijn directe collega's. En uitgeprinte GANTT kaart is een aantal A4 pagina's groot en staat collega's toe mee te kijken in mijn planning. Op deze manier wil ik draagvlak creëren voor het project, en indien collega's interesse willen tonen in een bepaalde ontwikkeling fase, dan nodigt zo'n kaart hiertoe uit.
- een GANTT kaart is sterk visueel opgezet. Ik heb dit versterkt door zelf op de kaart met fluocerende kleuren milestones aan te geven, en de voortgang aan te geven door de afgewerkte fases in te kleuren. Dit klinkt misschien kinderachtig, maar het versterkt het visuele effect van de kaart.
- Vooruitgang kan gevolgd worden in één oogopslag gevolgd worden door mijzelf, door collega's en door de begeleider.
- Duidelijke afhankelijkheden van de milestones en producten, volgens de AUP fasering in te richten.

Voor het inrichten gebruik ik de Open Source software Project Management[12], beschikbaar via het Debian software distributie systeem. Een eerdere poging om de software ProjectPlan HD voor de Apple iPad te gebruiken frustreert het planningsproces, omdat de invoer op een tablet niet ideaal is.

Zoals eerder vermeld (Hoofdstuk 4.5 Planning) heb ik de slack die in de globale planning aanwezig is, ook overgenomen in de detailplanning.



5-3 GANTT kaart met visuele markeringen

De volledige kaart is te vinden in de bijlage!

[12] Ook wel Planner genoemd, bron: http://live.gnome.org/Planner

6 Elaboration fase #1

Het doel van deze elaboration fase is om tot een ontwerp te komen. En om tot die stap te komen is, zoals bleek in de Inception fase, meer informatie nodig.

AUP geeft invulling aan de Elaboration fase op de volgende manier:

Modellering, uiteraard is de modellering in de ontwerpfasering erg belangrijk. Er moet tot een ontwerp gekomen worden, en dat ontwerp moet gemodelleerd zijn om het probleem op te lossen om zo de doelstelling te bereiken. Oftewel: Er moet een software tool komen die inconsistente data kan aan tonen volgens het IST SOLL model. Dat moet ervoor zorgen dat er geen financiële risico's gelopen worden.

Implementation, er moet een begin gemaakt worden aan de implementatie. Dit zou kunnen door het kiezen van een ontwikkelomgeving, maar ook door voorbereidend werk te doen, bijvoorbeeld bij het ontwikkelen van benodigde software libraries.

Testing, er moet een begin gemaakt worden aan het opzetten van een testomgeving. Dit zou kunnen via een set testeisen die na aanleiding van de bevindingen, eisen en wensen worden opgesteld. Maar dit zou ook een testomgeving in de software kunnen zijn, zoals bijvoorbeeld unit testing.

Deze fase duurt in totaal 6 weken, van week 2 tot en met week 8. Week 9 word effectief aan slack over gehouden.

6.1 Inventarisatie

Om tot een ontwerp te komen, is de eerste stap de inventarisatie en analyse van de beschikbare informatie relatief tot het onderwerp. In dit geval is er meer informatie nodig over het Intranet, waar integratie mee moet plaats vinden. Ook is er meer informatie nodig over de database, de netwerk apparatuur en de interactie met de eindgebruikers.

6.1.1 Intranet

De inventarisatie van het intranet bestaat uit een oriëntatie door de gebruikerskant. Dit leert dat het intranet een sombere interface is, waar met een HTML FORM/POST opzet data verwerkt word. Vrijwel ieder type data (zoal: klanten, hardware, datacentra, etc.) heeft een eigen pagina. Elke eigenschap kan bewerkt worden, en eigenschappen zoals gerelateerde tabellen linken direct door naar de betreffende pagina. Dit doet vermoeden dat er een relationele database achter zit.

Tevens bevat het Intranet een handboek, maar daarvan is mij reeds vermeld dat de informatie die daarop te vinden is, mogelijk sterk verouderd is.

De volgende logische stap is te kijken naar de broncode van het intranet. Het is voornamelijk geschreven in PHP, maar er word ook gebruik gemaakt van shell (bash) scripts die uitvoerbaar zijn onder PHP, of direct als CGI (Common Gateway Interface). Dat laatste word gebruikt om te communiceren met syslog bestanden op de management server en een switch beheer tool NAL.

Er word veelvuldig gebruik gemaakt van een standaard bibliotheek, met daarin functies om FORM velden weer te geven, en de HTML te renderen. De bibliotheken bestaan uit een mix van procedureel (sequentieel) inzetbare functies. Als deze software vanaf het begin van het bedrijf ontwikkeld is, en NLix is ondertussen 10 jaar actief, is het aannemelijk dat de basis gelegd is voor 2004, waarin PHP5 uitkwam (PHP.net, 2012). Ondersteuning voor object georiënteerde code was voor versie 5 summier.

Tevens is er een module aanwezig die tabellen genereert met data, zoals dat ook in bijvoorbeeld PHPMyAdmin gebeurt. Opvallend is dat deze module wel volgens de object georiënteerde manier geschreven is. Aangenomen kan worden dat deze niet door dezelfde persoon ontwikkeld is. Navraag bij mijn directe collega's verduidelijkt de situatie. Er is in het verleden een medewerker geweest die aan deze code ontwikkeld heeft, tot zijn grote frustratie word daarbij vermeld.

De frustraties zijn te snappen omdat er vrijwel geen documentatie in de code staat, referenties niet zijn gedocumenteerd en procedurele code schijnbaar lukraak word afgewisseld met object georiënteerde code.

De code die verantwoordelijk is voor het afhandelen van communicatie met de database bevat erg veel lange SQL strings. Er is geen filtering aanwezig voor de invoer en dezelfde code die de database communicatie afhandelt, zorgt ook voor de formattering van de uitvoer.

Navraag over de beveiliging leert dat het intranet als een vertrouwde omgeving word gezien. Inloggen is alleen mogelijk op basis van een whitelist en kan per sectie worden gedefinieerd. Over eventuele datacorruptie in de database word geen zorgen gemaakt, er worden continue back-ups van gemaakt en dat bij een foutje een rollback uitvoeren sneller is.

Als laatste valt mij op dat er binnen het Intranet verschillende controle tools bestaan, en die tools vrijwel allemaal niet meer gebruikt worden. Navraag levert op dat deze tools niet goed te

configureren waren, en vaak 'false-positives' genereerde. Dat leid dus tot het negeren van deze informatie. Het is dus belangrijk dat de oplossing geen onnodige meldingen gaat geven. Een andere functie die ontbreekt is het opschorten (postpone) van dergelijke meldingen, zodat deze niet meer worden weergeven.

6.1.2 Netwerkapparatuur

Het netwerkapparatuur word via een aantal verschillende systemen bijgehouden. Het intranet is er voor de inventarisatie van de hardware, waarbij er onderscheid gemaakt word tussen virtuele chasis als placeholder in de database, daar elke hardware unit vervangen kan worden, chasis, switch-blades, (fan) trays en netwerkpoorten.

Poorten zijn niet per definitie een onderdeel van de switch blade (ook wel fabric), en kunnen bestaan uit een aparte GBIC module. De poort zelf is, net zoals de virtuele chasis een administratieve entiteit.

In principe is vanuit het Intranet alle hardware te vinden, ook als deze nog in bestelling is of uit productie genomen is. Dit maakt het mogelijk gelijk in te zien over welke specificaties een bepaalde switch of router beschikt, en aan welke datacenter locatie deze gekoppeld is.

Naast het intranet word er met Cacti en Nagios de monitoring voor het netwerkapparatuur ingericht. Met deze heb ik weinig te maken, en ik krijg te horen van de beheerder van de monitoring Lucas van Schouwen dat de monitoring apart word bijgehouden, daar de metrics afhankelijk zijn per apparaat en klant. Typisch zijn bijvoorbeeld bandbreedte ge- of verbruik en BGP route informatie niet meetwaardes die terug te vinden zijn in de Intranet omgeving.

De eerdere interviews hebben opgeleverd dat er met de apparatuur gecommuniceerd word via:

- SNMP (Simple Network Management Protocol), in alle versies (1,2c en 3).
- Syslog, ter ondersteuning van de monitoring.
- NAL, voor het dagelijks beheer om snel in te kunnen loggen op de aparatuur.
- Telnet en SSH, soms ook 'out of band' via een GSM modem, als het een device niet beschikt over een directe verbinding met het internet.

Vermeld moet worden dat de inloggegevens voor deze apparatuur niet in de intranet database staat, maar dat er door Vincent samen met de mensen van Backbone een eigen database onderhouden.

Er word ook duidelijk dat er een grote diversiteit aan hardware actief is. Binnen de hardware van vergelijkbare types word er niet perse gebruik gemaakt van dezelfde hardware configuraties. Daar komt ook nog de complexiteit mee dat niet alle hardware voorzien is van dezelfde firmware of zelfs van hetzelfde besturingssysteem.

Globaal zijn er 3 typen systemen:

- Brocade (voorheen Foundry) switches en routers
- Juniper switches en routers (voornamelijk voor edge switches)
- MRV multiplexers en mediaconverters.

Er word mij door collega's vermeld dat niet alle communicatie standaard is. Zo is bij Brocade, ook als dit wel geschreven is in de handleiding, SNMP niet altijd correct geïmplementeerd. Er zijn wel MIB

(Management Information Bases) beschikbaar, maar dat garandeert nog niet een consistente reactie. Daar zijn Arno en Lucas vooral achter gekomen toen Cacti en Nagios ingevoerd werd, dat voornamelijk communiceert met SNMP.

Om het nog wat erger te maken, sommige SNMP velden kunnen ervoor zorgen dat het chassis effectief de hele SNMP daemon ophangen. Dit lijkt voornamelijk te gelden voor Brocade systemen, waarvan er binnen NLix veel gebruikt worden. Het is een bug, waarbij sensoren serieel op een bus zitten en er een single thread SNMP daemon draait. Brocade geeft dit binnen de eigen gebruikersgroepen maar mondjesmaat toe. Er word mij dringend geadviseerd niet zonder het backbone personeel in te lichten, ook maar iets te doen met fysieke sensoren, omdat het de monitoring en de daaraan gekoppelde facturatie kan problemen kan opleveren.

Een ander significante bevinding is dat de naamgeving van de commentaar velden in de netwerkapparatuur, zoals bijvoorbeeld bij de poortinformatie, niet consistent is. Het kan een nummer zijn, maar is in veel gevallen een afkorting van de klantnaam.

Deze bevindingen geven het vermoeden dat een universeel systeem creëren voor communicatie met *alle* netwerkapparatuur alleen mogelijk is door een grote mate van abstractie, of door beperking tot zeer specifieke, vastgestelde meetwaarden. Een vermoeden dat door mijn collega's mondeling keer op keer bevestigd word als ik met ze praat over de voortgang.

6.1.3 Database

De intranet database was het volgende onderwerp van inventarisatie. Allereerst moet er duidelijk gemaakt worden dat ik door middel van PHPMyAdmin volledig inzicht en toegang had tot de database, waardoor het systematisch in kaart brengen hiervan goed mogelijk is gemaakt. Er is door de opdrachtgever aangegeven dat er geen documentatie van de database beschikbaar is.

Omdat ik in het lezen van de broncode van het intranet al gezien had dat het om een relationeel model database gaat, is de eerste stap geweest de relaties in kaart te brengen. Dit voorkomt dat data dubbel word opgevraagd, of erger nog; dubbel word opgeslagen. Het dubbel opslaan van data is een ongewenste situatie omdat bij mutaties dan inconsistentie optreed. In dit geval is dat geheel in lijn met de IST SOLL gedachte die het bedrijf hanteert.

Ik besluit, om meer inzicht te krijgen over de interne database relaties, deze in kaart te brengen. Nu bied PHPMyAdmin standaard een beperkte optie om relaties visueel weer te geven. Echter deze functie kent de beperking dat dit alleen per tabel weergeven kan worden. Het is daarom per definitie niet geschikt voor deze taak. PHPMyAdmin zal wel een referentie vormen, ook in het verdere project.

Het word in PHPMyAdmin ook duidelijk dat ik te maken heb met een MyISAM database, en niet met een modernere INNODB database. MyISAM ondersteund geen zogenaamde Foreign Keys, en daarom moeten relaties hard in de code geprogrammeerd worden. In dit geval is dat gebeurd met het gebruik van ID velden. De opdrachtgever bied aan, dat als ik een overzicht kan creëren van de database, hij daar de relaties voor mij in kan vullen.

Om dit overzicht te creëren besluit ik gebruik te maken van een visualisatie tool om zodoende de database te 'reverse engineeren'. Ik stel dat betaalde software in eerste instantie afvalt, omdat er in principe geen budget voor dit project bestaat. Tevens is het lastig om gelicenceerde software te testen zonder eerste een vrij prijzige licentie af te nemen. Een internet zoektocht hiernaar bracht me,

naar de officiële MySQL ontwikkel omgeving: MySQL Workbench 5.2. Deze word onder een LGPL licentie vrijgegeven.

Onder Debian 6 is deze niet als binary verkrijgbaar. Bij de eerste compilatie poging blijkt dat er een flink aantal grafische bibliotheken ontbreken, iets wat naar alle schijn te danken is aan het gebruik van cross-platform software pakketten als GTK en Cairo2. Ik heb voor intern gebruik een script geschreven die alle afhankelijkheden installeert, de source bestanden download, deze compileert en de gebruiker naar de juiste directory stuurt. In bijlage 12.10 is het script terug te vinden.

MySQL Workbench is inderdaad in staat om de database velden in één overzicht te zetten en dit overzicht word door de opdrachtgever ingevuld (op papier). MySQL Workbench is alleen niet geschikt om deze relaties duidelijk weer te geven zonder het gebruik van Foreign Keys, wat inhoud dat de tabellen niet meer waarheidsgetrouw worden. Er moet dus een oplossing komen voor de visualisatie.

Ik besluit om een grafische bibliotheek in te zetten die ik in het verleden zelf veelvuldig heb ingezet om data te visualiseren, namelijk: GraphViz (Gansner & North, 2012). Het enige probleem is dat de data dus op papier stond, en niet digitaal beschikbaar was.

GraphViz werkt met een specifieke opmaaktaal genaamd DOT (Gansner & North, 2011). Wat een simpele syntax is dat vier type data kent:

- Nodes (velden)
- Edges (verbindingen)
- Graphs (clusters)

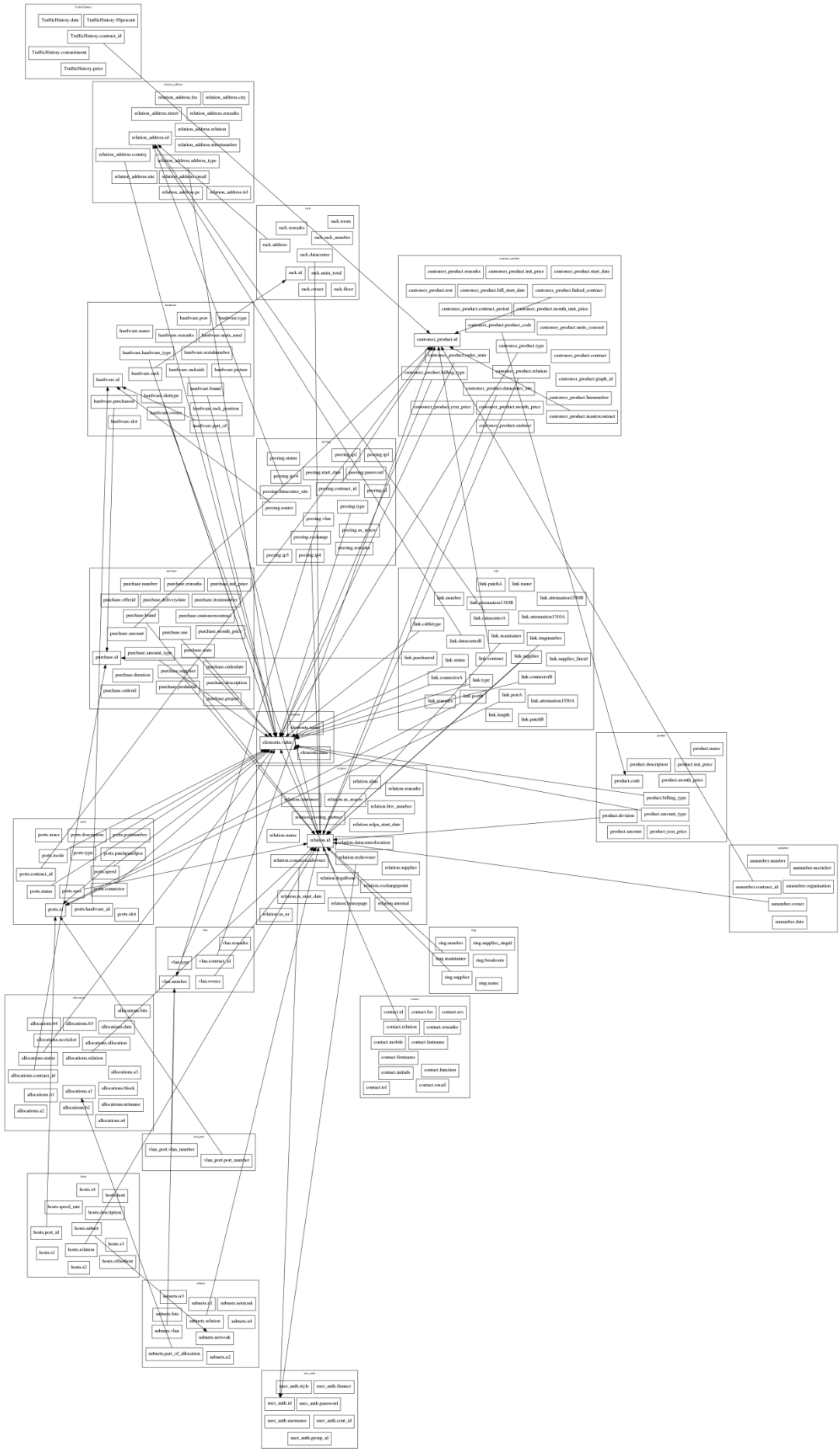
Ik heb in het verleden gebruik gemaakt van een PHP wrapper die PHP objecten automatisch vertaald in de DOT taal. Deze wrapper is onderdeel van de PHP PEAR bibliotheek en is verkrijgbaar via: http://pear.php.net/package/Image_GraphViz . Deze wrapper staat mij toe om de database informatie in een redelijk simpele vorm in te voeren als PHP array, en de links te definiëren, zoals deze door de opdrachtgever aangegeven zijn. GraphViz handelt vervolgens het genereren van de visuele renders af.

Het gebruikte script is te vinden als bijlage in het Elaboration document, hoofdstuk 8.6.

Het resultaat is te vinden op de volgende pagina. Het is op A4 niet praktisch leesbaar, maar het geeft een indicatie van de complexiteit.

De visuele weergave heeft gedurende het project gediend als referentiekaart en als communicatiemiddel richting de opdrachtgever en mijn collega's.

6-1 Resultaat visualisatie database



6.2 Opzet architectuur

Na het inventariseren van de omgeving, werd er begonnen aan het opzetten van de architectuur.

De Inception fase gaf al een aantal belangrijke ontwerpoverwegingen aan:

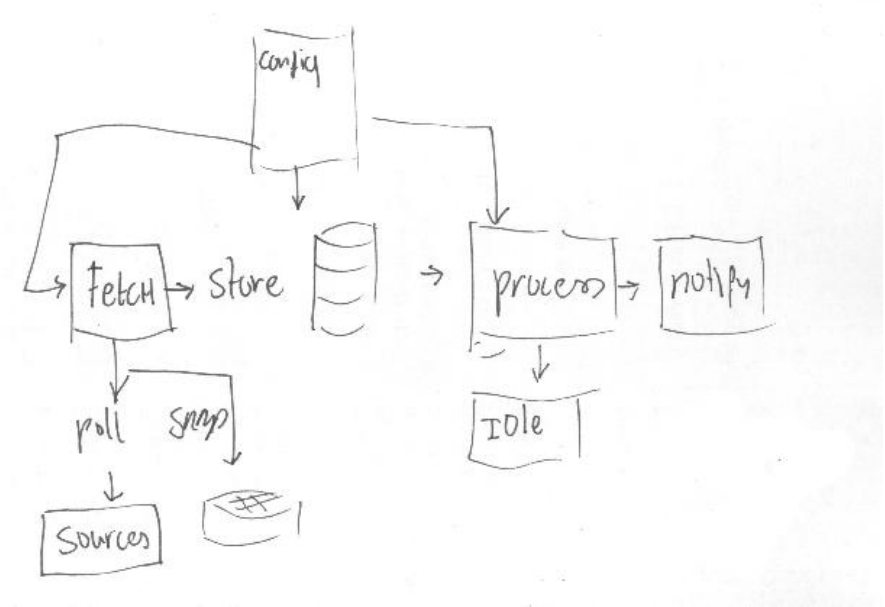
- De oplossing detecteert of IST wel overeenkomst met SOLL, feitelijk word er gekeken of er een status van synchronisatie bestaat tussen de twee testwaarden.
- De oplossing zal moeten bestaan uit verschillende modules:
    - Een intranet module, voor de gebruiker interactie
    - Een commandline gebaseerde module, voor communicatie met het netwerk
    - Een rapportage module, voor de notificaties via de email.
- De oplossing zal configureerbaar moeten zijn.
- De structuur moet schaalbaar zijn, en uitbreidbaar.
- Er moet draagvlak zijn voor het gebruik van de software.
- Historische gegevens, ter inzage is een wens.
- De correcte SOLL naar IST pushen, dit is een wens.

Daar kan nu aan worden toegevoegd:

- Foutieve meldingen (false-positives) worden niet gewaardeerd, en waarschijnlijk genegeerd.
- Er zal, vanwege de enorme diversiteit op het netwerk gekozen moeten worden voor of een zeer abstracte configuratie, of voor een kleine dataset.
- Het opschorten van meldingen, voor bijvoorbeeld een switch, moet mogelijk zijn.

Verder word in direct overleg met collega's een lijst aangelegd met specifieke datavelden van informatie die zowel in de database als in het netwerk beschikbaar is, op in ieder geval het switch platform. Dit om te voorkomen dat het ontwerp onnodig complex word.

Dit leverde de eerste ontwerpschets op:



6-2 Eerste ontwerp

De keuze voor de ontwerptaal is Engels omdat dit gangbaar is binnen het bedrijf.

Opvallend is dat de keuze gemaakt is de tool op te delen in modules. Dit verbetert wat mij betreft de leesbaarheid en de onderhoudbaarheid van de software. Tevens forceert dit duidelijke definities van de in- en uitvoer van de individuele modules, iets wat de documentatie alleen maar ten goede komt.

Omdat dit geen traditionele software is, zal er geen continue loop bestaan, zoals een Main() loop veel gebruikt word in C-achtige omgevingen. De initiatie van het IST SOLL proces zal geïnitieerd worden door de crontab daemon die actief is op de intranet server. De stappen zijn als volgt:

1. Cron deamon roept de tool aan.
2. De tool leest de configuratie in (met daarin de database informatie).
3. De module die de IST uitleest haalt informatie uit de database om te verbinden en kijkt per netwerk apparaat (via de CLI, NAL of SNMP) welke meetwaardes uitgelezen moeten worden.
4. De meetwaardes worden opgeslagen in de database, voor historische doeleinden.
5. De module die SOLL uitleest (uit de database) haalt deze gegevens uit de database en
6. De processing module legt de IST en SOLL naast elkaar en constateert dat er wel of geen sprake is van synchronisatie. Indien er synchronisatie is word er geen actie ondernomen.
7. Indien er geen synchronisatie is, maar het apparaat staat aangemerkt als in onderhoud, dan zal er geen notificatie in de buffer terecht komen. In plaats daarvan word deze vermeld in de interface, kolom: erkende notificaties.
8. Indien er geen synchronisatie is zal er een notificatie in de buffer komen.
9. Zodra alle objecten getest zijn worden de notificaties uit de buffer verzonden per email naar het NOC.

De interface zal een drietal kolommen moeten bevatten:

1. Een kolom met actieve meldingen.
2. Een kolom met meldingen die gemarkeerd zijn als erkende notificaties.
3. Een kolom met historische meldingen.

Er zal een abstract model gevormd worden van de IST en SOLL meetwaarden. Er zal een model gevormd worden om de rapportages in te kunnen verwerken, er zal een database model moeten komen voor database interactie en er zal een model moeten komen voor communicatie de netwerkapparatuur.

Het voordeel van deze abstracte manier van modellering is dat op deze manier het testen versimpelt word, en het is tevens mogelijk om geheel in lijn met AUP, unit testen mogelijk te maken. Immers als elke module eigen functionaliteit bevat, en dat kan bevestigd worden met een unit test, mag er in de rest van het ontwikkelproces worden aangenomen dat het werk van die betreffende module goed word afgehandeld.

6.3 Keuze ontwikkelmethodiek

De keuze om voor een object georiënteerde methode te kiezen ligt voor de hand, want:

- De code moet goed onderhoudbaar zijn.
- De code moet schaalbaar zijn.
- Er moet aan ontwikkeld kunnen worden door andere engineers.
- Het bied veiligheid bij database interacties, door objecten te scheiden van SQL.
- Het bied veiligheid bij netwerk interacties, door data te scheiden van de communicatielaag.
- Ik ben ermee bekend, en hiermee opgeleid.

Er zijn ook een aantal nadelen aan de keuze:

- Het word momenteel slechts beperkt toegepast binnen NLix.
- Initiële prototyping is wat lastig, omdat er tijd nodig is een framework op te bouwen. Er is dus een langere aanlooptijd nodig. Deze is echter al in de planning opgenomen en moet geen belemmering vormen.

De nadelen zijn ook gelijk de voordelen van het procedureel programmeren, namelijk het word momenteel actief toegepast en het prototypen is sneller aan te vangen. Wat mij betreft wegen de nadelen niet tegen de voordelen op en gaat de keuze uit naar een object georiënteerde taal. Bij het kiezen van een programmeertaal heb ik rekening gehouden met mijn persoonlijke limieten. De eisen die aan de programmeertaal gesteld worden zijn:

1. Ik moet er voldoende mee bekend zijn.
2. Het moet goed integreerbaar zijn met MySQL.
3. Het moet effectief inzetbaar zijn op de huidige Intranet server (Linux).
4. Het moet toegang hebben tot de shell om te communiceren met NAL.
5. Het moet toegang hebben tot SNMP bibliotheken.

Van de volgende talen heb ik kennis genoeg om een tool te ontwikkelen. De keuze bestaat uit:

- PHP5 Object georiënteerde notitie
- C++
- C#

De keuze is als volgt:

	Mee bekend	MySQL	Linux	Shell	SNMP	**Totaal**
PHP5	++	++	++	+	++	**++**
C++	+	-	++	++	+	**+**
C#	-	++	--	-	+	**-**

Tabel 2 Keuze programmeertaal

PHP5OO is de geschikte keuze. Het zwakke punt van C# is dat het ontwikkeld is door Microsoft als een multiplatform omgeving, maar initieel niet actief bibliotheken heeft ondersteund voor Linux. Een deel van de source code is geëmuleerd vanuit .NET en draait niet optimaal op Linux. Verder is het uitrollen van Mono op een productie server niet aan te raden. C++ is een geschikte optie, maar de integratie met MySQL is relatief ten opzichte van PHP complex en minder ondersteund. PHP bied het

voordeel van de huidige ondersteuning vanuit NLix en de uitgebreide documentatie die online beschikbaar is.

6.4 Modellering

Na aanleiding van de bevindingen ben ik aan de slag gegaan met het maken van de modellering.

Voor de modelleringstaal heb ik gekozen voor de industrie standaard UML (Unified Modelling Language) met het doel de systeem architectuur inzichtelijk te maken voor de betrokken partijen. Dit sluit aan bij de richting van AUP. Het toepassen van de iteratieve ontwikkelingcycles in AUP staan mij toe iets minder detail in de modellering te stoppen om zo meer ruimte te geven voor input bij het ontwikkeltraject door de opdrachtgever en eindgebruikers.

Het probleem waar ik gelijk tegen aan liep is dat het met UML lastig is te bepalen waar je moet beginnen. Die realisering werd gevolgd door het besef dat er niemand binnen NLix was die me hierbij kon helpen. Om dat probleem op te lossen, en om zeker te weten dat ik UML modellering volgens de standaard kan hanteren, ben ik de boeken in gedoken.

Ik heb het boek Pragmatisch modelleren met UML 2.0 van Sander Hoogendoorn (Hoogendoorn, 2011) als uitgangspunt genomen, daar deze door medestudenten en door internetgebruikers vermeld werd als één van de betere werken op dit gebied.

Het boek leerde mij dat UML, naast modelleringstaal ook een functie moet hebben als communicatiemiddel met de opdrachtgever. Het zou wel eens kunnen zijn dat UML helemaal niet per definitie het meest geschikte model is om te communiceren met een opdrachtgever die graag mondeling overlegt en abstracte diagrammen niet perse interessant vind.

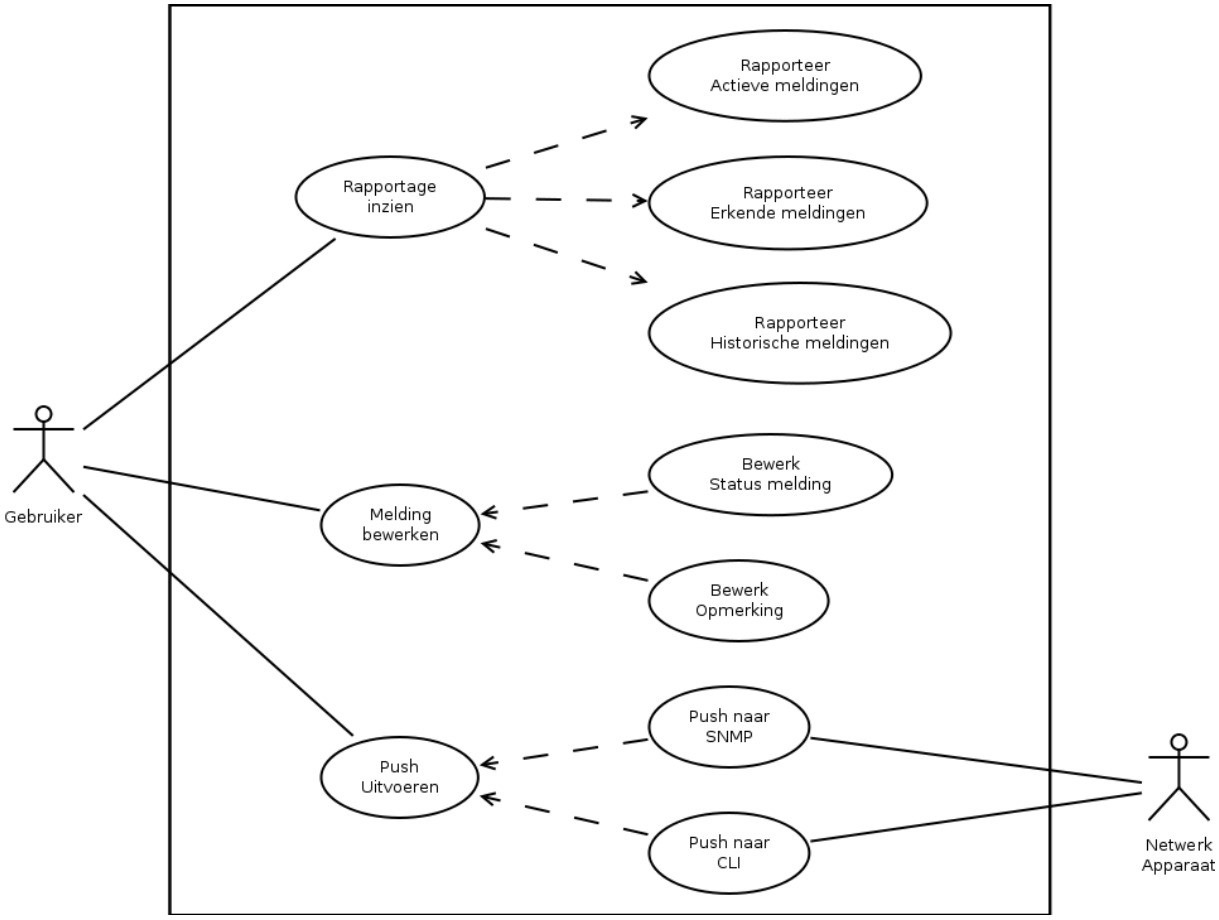
6.4.1 Software

UML modelleren is één ding, dit vastleggen in diagrammen is een heel ander probleem. UML is ontworpen om visueel herkenbaar te zijn, dus het medium mag niet uitmaken. Maar vanwege de formele rapportage aan de opdrachtgever is het niet acceptabel om schetsen op papier in te leveren. Goede UML modelling software loopt aardig in de papieren, maar er zijn Open source alternatieven op de markt. Sommige van deze pakketten claimen dat als de diagrammen goed genoeg ingevoerd worden, dat de code er letterlijk uit komt rollen.

Vanwege de tijd die ik besteed heb met het inlezen over UML, heb ik minder tijd kunnen besteden aan de keuze van UML software. Ik heb een aantal Open Source pakketten uitgeprobeerd, maar zonder uitzondering hebben deze pakketten te maken met een zeer onvriendelijke gebruikers interface die niet vanzelf spreekt en velden niet toelicht. Dit maakt het modelleren onnodig complex in mijn ogen. Voor de schema's heb ik gebruik gemaakt van het Linux Visio alternatief: Dia.

6.4.2 UML uitwerking

Ik heb ervoor gekozen met de procesdiagrammen te beginnen. Als het proces inzichtelijk is kunnen de use-cases opgezet worden. Deze zijn direct afleidbaar van de eisen, wensen en aanvullende bevindingen.

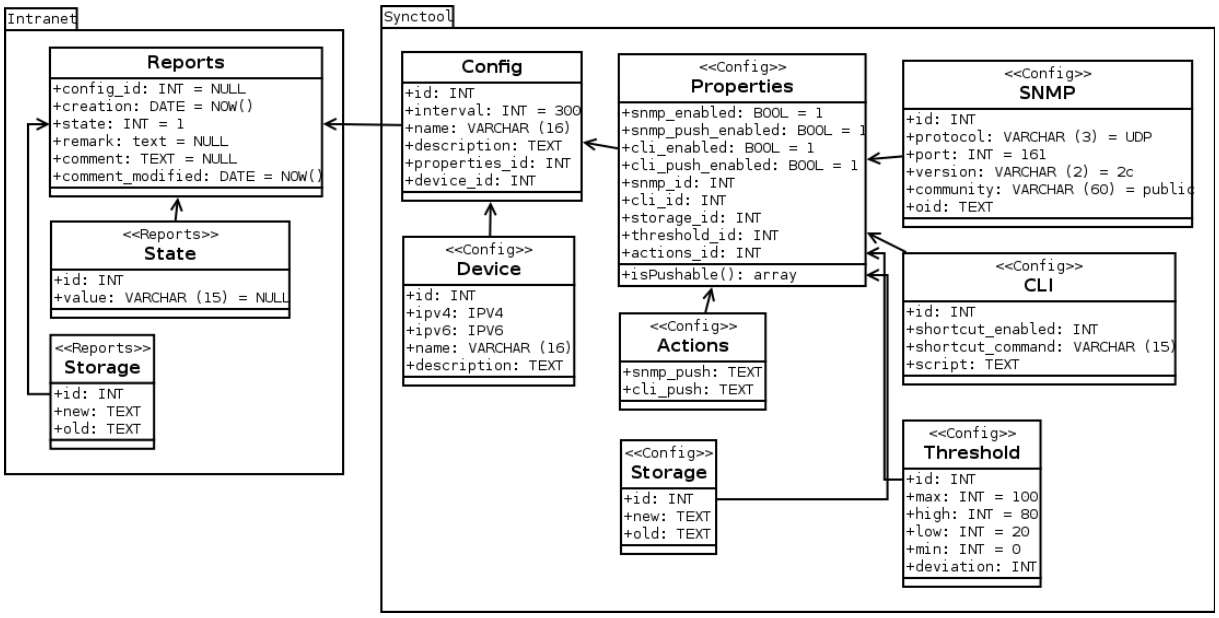


6-3 UML use-case voor het Intranet

Zodra ik die eisen en wensen verwerkt had was ik ervan overtuigd dat het voldoende is. Het UML boek heeft mij geleerd dat use-cases per definitie niet volledig zijn, omdat het niet alle acties beslaat. De gedefinieerde actoren vanuit het use-case perspectief zijn niet per definitie alle onderdelen waar het systeem mee te maken krijgt. Achtergrondprocessen zoals database interactie worden hier volledig overgeslagen.

Zowel de procesdiagrammen en use-cases zijn onderdeel van het Elaboration document, dat als bijlage onderdeel van dit document vormt.

Vanuit het ontwerp zijn er ook verschillende klasse diagrammen opgesteld, hierin word reeds rekening gehouden met het feit dat de datamodellen opgeslagen gaan worden in MySQL.



6-4 Klassediagram uit Elaboration document

6.5 Afronding

Aan het eind van de Elaboration fase word er de tijd genomen alle bevindingen, diagrammen en informatie te verwerken tot het Elaboration document, als rapportage naar de opdrachtgever en als materiaal voor de milestone bespreking.

6.6 Procesbewaking

De planning stelt dat het Elaboration document ingeleverd word in de 9e week van het project. Echter omdat er meer tijd besteed moest worden aan het verdiepen in UML heeft dit één extra week gekost. Dit valt precies binnen de slack die er voor de Elaboration fase gesteld is.

Het Visie document stelt dat het Elaboration document het volgende dient te bevatten:

- Bevat een goedgekeurd software eisenpakket.
- Bevat een duidelijke modellering van de uit te werken software.
- Bevat alle beschikbare informatie die gebruikt kan worden om software
- te ontwerpen.
- Bevat informatie die gebruikt kan worden om functionele testeisen op
- te stellen.
- Dient gevalideerd te worden door de opdrachtgever.

De proceskwaliteit eis:

- Er is tijdens het opstellen voldoende tijd en aandacht besteed aan het onderzoek om de definitieve eisen en wensen te kunnen verantwoorden.

In alle gevallen, behalve de validatie door de opdrachtgever op het moment van inleveren, zijn hiermee beschikbaar.

7 Construction fase - Sprint cycle #1

In afwachting van feedback op het ontwerp ben ik in week 11 begonnen met de constructiefase. Deze fase duurt volgens de planning 4 tot weken. Daar ik van mening was dat het ontwerp de visie van de opdrachtgever duidelijk reflecteerde, en ik daar alle informatie in verwerkt had, vond ik het een acceptabel risico om te wachten op feedback. Zeker omdat dit in het visie document ook bijna twee weken heeft geduurd. Twee weken niks doen op een project van effectief 17 weken is geen optie.

De Construction fase word gekenmerkt door:

- Het uitwerken van de software tot het punt waar deze klaar is voor het testen.
- Het uitwerken van de architectuur en het ontwerp door middel van sprint cycli.
- Het unit testen van de software.

De invulling van de disciplines binnen de Construction fase ziet er in AUP als volgt uit:

- **Model**
  - Het afronden van de modellering
- **Implementatie**
  - Bouwen aan de onderliggende bibliotheken
  - Het unittesten van bibliotheken
  - Het bouwen van de logica.
  - Het bouwen van de gebruiker interface
  - Koppeling met externe systemen (legacy)
- **Test**
  - Het uitvoeren van (unit-)testen
  - Het zonodig updaten van het testplan
- **Deployment**
  - Ontwikkel de productieomgeving
  - Ontwikkel de installatiescripts
  - Werk de 'release notes' uit.
- **Configuration Management**
  - Het continue bijwerken van de ontwikkel omgeving
- **Project management**
  - De planning waar nodig updaten
  - De risico's beheren.
  - Het afsluiten van de fase dmv communicatie met de opdrachtgever.
- **Environment**
  - Communicatie met collega's verantwoordelijk voor de productie omgeving.

De Construction fase sluit af met het opleveren en de goedkeuring van de software voor productie. Tevens moet er in deze fase testrapportage gegenereerd worden om de productie geschiktheid van de software te kunnen garanderen.

7.1 Inrichting werkomgeving

De werkomgeving waarin de code ontwikkeld word, moet tijdens de Construction fase opgezet worden. Er zijn een aantal afwegingen die ik moet maken om voor een effectieve omgeving te zorgen.

De overwegingen die gemaakt zijn is het wel of niet gebruik maken van een versie systeem, de ontwikkel editor en het uitrollen van de (test)omgeving.

Afweging: Wel of geen versiebeheer systeem gebruiken?

Het gebruik van een versiebeheer systeem als CVS, SVN of GIT bied de volgende voordelen:

- Terugrollen is simpel, omdat de veranderingen per wijziging worden opgeslagen en alle voorgaande versies ingezien kunnen worden.
- Het uitrollen van de software is versimpeld.
- Het onderhoud van de software is beter te managen.

De nadelen zijn:

- Het is een extra stap in het onwikkelproces, elke wijziging moet worden vastgelegd en ge-commit worden, wat niet ideaal is voor rapid prototyping.
- Omdat er geen versiebeheer systeem aanwezig is bij NLix zal deze eerst opgetuigd moeten worden.
- Er is geen intentie bij collega's om een versiebeheer systeem te leren, dat dient de onderhoudbaarheid dus niet.
- De keuze van versiebeheer systeem is niet simpel, het is erg aan persoonlijke voorkeur gebonden bij ontwikkelaars.

Er zijn meer nadelen die niet opwegen tegen de voordelen van een versiebeheer systeem. De keuze word gemaakt op lokaal niveau bij elke grote wijziging de code in een nieuwe folder te plaatsen, om op deze manier wel terug te kunnen kijken naar oude code.

Afweging: Keuze ontwikkel editor

Omdat er vanuit NLix niet een standaard taal of editor vereist word, en ik onder een voor mij relatief nieuw systeem werk, heb ik nog geen idee welke editor ik wil gebruiken. Mijn minimal eis is dat de editor beschikt over het zogenoemde syntax highlighting. Bij voorkeur moet PHP5 daar ook bij ondersteund worden, maar C style is ook acceptabel. Ik heb de volgende opties:

- De standaard Gnome editor Gedit. Ondersteund PHP5 syntax highlighting
- Eclipse PHP. Een Open Source editor met PHP5 syntax highlighting. Heeft een overkill aan features en is voornamelijk gericht op Java ontwikkeling

Bij deze afweging komt kijken of ik wel of niet gebruik wil maken van de IDE (Intregrated Development Environment) mogelijkheden van Eclipse. Het verschil zit hem in het feit dat een IDE een eigen lokale testomgeving heeft draaien. Echter, als deze testomgeving niet 100% identiek is aan de productie omgeving, zou dit op een later moment problemen kunnen opleveren. Omdat ik niet eis

dat er een directe database verbinding nodig is en ook geen versiebeheer software intregratie acht ik het onnodig een IDE in te richten en leren te gebruiken. Een simpele tekst editor voldoet aan de eis.

Uitrollen van de code

Het enige wat hier nog overblijft is hoe de code uitgerold gaat worden. De enige uitrol die er gedaan moet worden om de code vanaf de werkstation naar de server te krijgen is deze op de juiste directory op de Intranet server te plaatsen. Ik krijg toestemming van de server beheerder Lucas om in een subfolder (/synctool) aan de slag te gaan, zolang ik geen andere elementen van het Intranet stoor.

Er komt een script dat bij het uitvoeren, door gebruik te maken van SCP[13] de bestanden van het werkstation naar deze map kopieert. De enige belemmering was dat het bij elke keer uitvoeren nodig was om het wachtwoord uit te typen nodig was. De oplossing is het veranderen van de authenticatie methode van wachtwoord naar public key. Door het genereren van de public key, en deze toe te voegen aan de lijst met geauthoriseerde clients op de server word geheel automatisch de beveiligde verbinding opgezet en worden de bestanden automatisch overgeschreven. De uitvoerbare code staat hieronder vermeld:

```
# Sync over SCP
/usr/bin/scp -r /home/gerben/ Afstuderen/Code/synctool/* gerben@intranet.nl-
ix.net:/var/www/intranet/synctool
```

Ik heb totaal 2 dagen besteed aan het correct inrichten van de werkomgeving en het werkend krijgen van het uitrol script.

Na het treffen van de voorbereidingen en nu de werkomgeving ingericht is, werd het tijd te starten met de eerste spring cycle.

7.2 Framework

De eerste stap die genomen word is het opzetten van het framework ter ondersteuning van de code. In dit framework gaat een aantal modulles ingezet worden, namelijk:

- Database module
- SNMP module
- CLI (CommandLine Interface, ook wel shell) module
- Mail module
- Intranet module
- Ontwikkel (debug) module
- Configuratie module

Ik breng het initiatieproces van deze modules onder in een Main() programma wat vanaf verschillende interfaces kan worden aangeroepen. Enerzijds is er de Intranet webinterface, die via de PHP CGI word aangeroepen en HTML renderd. Anderzijds is er de PHP CLI die via de CLI word aangeroepen en pure tekst teruggeeft.

[13] Secure Copy, een data protocol voor SSH (Secure Shell). Waarbij de gebruiker data over een standaard beveiligde verbinding kan overzetten.

Ik realiseer me al snel dat de uitvoering van dit ontwerp de vormen aanneemt van een MVC framework, wat mijns inziens een goede ontwikkeling is.

7.2.1 Model, Viewer and Controller.

Het Model, Viewer and Controller model is gebaseerd op het idee dat er drie verschillende entiteiten zijn binnen een stuk software.

- Model - Dit stuk is verantwoordelijk voor de modellering van de objecten. Dit is meestal een klasse definitie met de bijbehorende functies en functionaliteit. Ook worden er de variabelen gedefinieerd en de accepteerbare invoer gedefinieerd.
- Viewer - Dit is een stuk feedback naar de gebruiker. Al kan een view ook een zogenaamde API (Application Programmable Interface) zijn waar andere software direct kan communiceren met de te ontwikkelen software.
- Controller, is de hoofdmodule van het MVC model. Dit stuk code is verantwoordelijk voor de systeem logica. De controller handelt ook de aanvragen van de gebruiker af, haalt waar nodig Modellen op, vormt de reactie daarop met de gegeven logica en geeft de gebruiker een View terug.

MVC staat op deze manier toe om verschillende interfaces te hebben aan één stuk software. Tevens kunnen er verschillende modellen samenkomen, en blijft op die manier de communicatie tussen de software en bijvoorbeeld het netwerk gescheiden.

Een bijkomend voordeel is dat het een uitermate geschikt model is voor een object georiënteerde stijl van programeren. Het forceert een duidelijke scheiding tussen de modellering, interfaces en visuele renders.

MVC dicteert niet hoe interfaces ingericht moeten worden, maar het dicteert wel dat interfaces zoals databases duidelijk gescheiden moeten zijn van de code. Het kan dus niet zo zijn dat SQL code terugkomt in de controller. Dit sluit wat mij betreft perfect aan bij het streven veilig met de database om te springen. Dit voordeel is ook toepasbaar op het netwerk, er kan geen directe code voor interactie met het netwerk instaan. Er moet dan wel voor gezorgd worden dat de interfaces expliciet definiëren wat wel en niet acceptabele invoer is, en daar moet op gecontroleerd worden.

Het enige, en misschien wel grootste nadeel van MVC in deze situatie is dat het debuggen lastig kan zijn. Dit omdat code nu dynamisch gecombineerd word, en dat ongedocumenteerde resultaten op kan leveren bij prototyping. Dat is mogelijk omdat niet alle combinaties van code reeds gedocumenteerd zijn.

7.2.2 Autoload

PHP5 OO heeft een speciaal mechanisme om automatisch en dynamisch benodigde bibliotheken in te sluiten. Dit is een optimalisatie en werkt erg goed ter ondersteuning van het MVC model. Dit mechanisme word AutoLoad genoemd. Dit doet precies wat het suggereert, het laad automatisch klassen aan die aangeroepen worden. Het is mogelijk om 2 type te gebruiken, het standaard autoload mechanisme van PHP, of het uitgebreidere SPL systeem.

Standaard autoload accepteert alleen de naam van de klas, terwijl het SPL_autoload_register systeem ook prependen en het errorniveau toestaat. Dat laatste is erg handig omdat het niet altijd gewenst is dat er een error conditie plaats vind zodra een klasse niet gevonden kan worden. De doorslaggevende factor was het feit dat de standaard autoload niet werkt bij het gebruik van de PHP CLI omgeving.

Ik heb er uiteindelijk voor gekozen om het SPL systeem te implementeren. Hieronder staat de sectie uit de code die ik daarvoor gebruikt heb:

```
/* Autoloading - Automaticly loads classes following this syntax:
 * project_path/class/example.class.php
 *
 * Notice: Traditional autoloading is not used because PHP indicates it as
 * discouraged [http://php.net/manual/en/language.oop5.autoload.php].
 * Secondly, autoloading is not available in PHP CLI mode, rendering it
 * useless as a commandline application.
 */
function php_autoload ($class) {
    $file = __DIR__ . "/class/" . strtolower($class) . ".class.php";
    if(file_exists($file))
    {
        decho("Autoload: Including [$file]..");
        require_once($file);
    }
    else
    {
        throw new Exception("Unable to locate [$file].");
    }
}
spl_autoload_register("php_autoload");
```

7-1 PHP code SPL Autoload

Opvallend hier is dat er gebruik gemaakt word van de functie decho(). Wat een functie is die conditioneel de echo weergeeft. De conditie is dan of de software wel of niet in debugging modus draait.

De exceptions worden op het moment dat deze code geschreven word alleen afgevangen door de standaard PHP Exception Handler, maar hier zal nog verandering in moeten komen.

7.2.3 Error handling

Ik liep tijdens het ontwikkelen van de Autoload code direct aan tegen een probleem, namelijk de error handling van PHP. Tijdens het testen van de autoload code kwamen er geen errormeldingen op het scherm toen er gebruikt werd van de PHP CGI (webinterface), maar wel tijdens het gebruik van de PHP CLI (shell). Nu is het goed mogelijk dat in de productieomgeving er gebruikt gemaakt word van twee verschillende standaard configuraties, maar ik had geen rechten om deze in te zien.

Ik heb geprobeerd om de standaard waardes te overschrijven met de zogenaamde php_ini_set() functionaliteit, waarmee de standaard waardes overschreven kunnen worden. Echter dit blijkt conditioneel, en ik heb hier geen rechten toe.

Na het zoeken op internet fora naar een oplossing van dit probleem, vond ik informatie die suggereerde dat het gebruiken van een eigen error handler het probleem op kan lossen. Om dit te bewijzen heb ik gebruik gemaakt van de set_error_handler functionaliteit.

```
//Custom Error handling (because its silenced by default in CGI)
set_error_handler("catchError", E_ALL);
```

7-2 PHP code Error handler

Deze handler krijgt van elke error melding de volgende informatie:

- Level (Type error)
- Message (Error melding)
- File (bestandsnaam)
- Line (regelnummer)
- Context (textuele melding)

Door zelf een functie (catchError) te schrijven die deze informatie afhandelt, en afhankelijk van de PHP CLI of CGI een uitvoer geeft met deze informatie. Dit bied mij ook de mogelijkheid selectief te zijn over welk niveau van meldingen er op het scherm zichtbaar moet zijn, ik besluit de types: Warning en Notice voor productie uit te sluiten, maar voor de ontwikkeling toe te laten.

7.2.4 Configuratie en debugging

Ik besluit om de configuratie op een globaal niveau vast te leggen, dit zodat iedere module toegang heeft tot informatie van bijvoorbeeld de uitvoerpaden, debug status en database informatie.

De configuratie word door de Main() klasse ingesloten als lokale variabele zodat klasses die aangeroepen worden, ten alle tijden de configuratie op kunnen vragen via de Parent klass Main().

Aanvullend op de configuratie is er een aparte set debugging functies. Deze functies moeten de interne uitvoering van informatie tijdens het ontwikkelen inzichtelijk kunnen maken. Deze functies worden in een apart bestand (debugging.php) geplaatst en bevat functies als de eerdergenoemde decho() en de krachtige functie d(). De laatste geeft vrijwel elk type ingegeven informatie structureel weer, waarbij er duidelijk word wat voor datatype de informatie is, en welke informatie deze bevat.

Standaard doet PHP niet aan een zogenaamde stacktrace, door een stacktrace functie toe te voegen kan deze nuttige data nu binnen het framework benaderd worden.

7.3 Interfaces

Nu het framework klaar is, is het tijd de onderliggende bibliotheken en de klasse vorm gaan geven. Een onderdeel van deze bibliotheken en klasse zijn zogenaamde interfaces, deze stukken code kunnen zowel een Model als Controller zijn, afhankelijk van de gebruikte programmerstyle, daar sommige klasse ook Controller functionaliteit bevatten.

7.3.1 PAL For NAL

De eerste stap om te communiceren met het netwerk is de commandline. Dit kan zowel via Telnet als via SSH, maar de voorkeur binnen NLix word gegeven aan Telnet, daar dit de standaard configuratie is en geen 'gezeur' oplevert met certificaten. Tevens word SSH standaard niet door alle apparatuur ondersteund.

Voor netwerk communicatie kan ik gebruik maken van het NAL (Network Abstraction Layer) script. Op deze manier hoeft mijn ontwerp niet apart gebruik te maken van de database met login informatie, die intern het bedrijf als erg gevoelig word beschouwd.

Omdat NAL niet direct geschikt is als API, verzoek ik Vincent Bourgonjen de oorspronkelijke code zo te wijzigen dat het mogelijk word om alleen de direct relevante tekst op het scherm terug te krijgen, in plaats van de gehele login sequentie. Deze heeft het erg druk, en ik besluit zelf een filter te schrijven.

De filter haalt alleen maar informatie van het scherm af als de lijn begint met een # hashtag, wat aangeeft dat het apparaat in Enabled modus staat. Vanuit Enabled modus is de meeste informatie uit te lezen, en kan er vanaf dat moment ook de Config modus aangezet worden.

Een functionaliteit van NAL waar ik op gewezen word is de mogelijkheid een lijst met commando's in te geven, in plaats van een enkele commandline functie. Dit laatste is erg nuttig als ik voor één device meerdere datavelden wil uitlezen, dit om te voorkomen dat er meerdere malen verbonden moet worden met een device, iets wat alleen maar overhead oplevert. In sommige gevallen zit er op netwerk apparatuur ook een limiet aan hoeveel sessies er actief mogen zijn, in het ergste geval zou het script dus een engineer van het netwerk device af schoppen.

Dit is uiteraard niet de bedoeling, dus ik besluit ook de lijst functionaliteit in de bouwen, ik maak een functie die een array van waardes accepteert, deze converteert naar een tekstuele lijst, deze lijst vervolgens in een tijdelijk tekstbestand opslaat en uitvoert. De resultaten worden dan als één geheel terug gegeven.

Het uiteindelijke resultaat is de CLI klasse, met de bijnaam: PAL (PHP abstraction Layer) for NAL.

Hier onder staat een klein stukje CLI klasse code om het gebruik van commentaar weer te geven.

```
class CLI
{
    public $status;     // (int) Return status (int) NAL commandline
    public $output;     // (array) Output from NAL command
    public $return;     // (string) Return status from NAL commandline
```

7-3 PHP code CLI klasse PAL

7.3.2 SNMP in PHP5

De tweede mogelijkheid om met de netwerk devices te communiceren is via SNMP.

SNMP is het Simple Network Management Protocol. Een IP gebaseerde client-server oplossing die beheerders toestaat, indien geautoriseerd informatie op te vragen en te wijzigen op de server. Dit is een industrie standaard in ICT, waar het vooral toegepast word voor configuratiebeheer en monitoring.

PHP kent een standaard implementatie voor SNMP, die gescheiden is per versie, te noemen versie 1, 2c en 3. Elk van deze versies kent eigen authenticitie methodiek en er zijn aparte functies voor.

De potentiële moeilijkheid hier is dat voor het gebruik van SNMP, voordat de klasse ingezet kan worden, alle informatie al beschikbaar moet zijn, namelijk de authenticatie gegevens, de locatie van de op te vragen informatie en indien de SNMP server op een niet standaard poort draait, zal die informatie ook bekend moeten zijn.

Ik besluit de SNMP functionaliteit op te splitsen in twee stukken:

- SNMP() als klasse voor SNMP (socket) interactie
- SNMPCommands() als klasse voor het uitvoeren van een specifieke informatie query.

Om voor SNMP() alle gebruikte velden te vinden heb ik gekeken naar welke invoer SNMP PHP accepteert en ik heb de RFC (Request For Change) documenten gelezen waarin dit voor de industrie gedefinieerd word. Tevens heb ik gekeken naar een voorbeeld implementatie die veelvuldig in de industrie gebruikt word, namelijk Cacti.

Ik kom tot de conclusie dat de volgende velden relevant zijn:

- MIB (Management Information Base), de locatie van het bestand met specifieke informatie betreffende informatie queries die niet standaard gedefinieerd zijn.
- IP Protocol (TCP of UDP)
- IP Poort (nummer)
- Versie, versie 1, 2c of 3, belangrijk om de authenticatiemethode en te gebruiken klasses te kunnen bepalen
- Community, ook wel de authenticatiestring voor versie 1 en 2c
- Host, het IP host adres
- OID, Object IDentifier
- Gebruikersnaam, voor authenticatie in versie 3
- Password, voor authenticatie in versie 3
- Auth_protocol, voor authenticatie in versie 3
- Auth_level, voor authenticatie in versie 3
- Priv_protocol, voor authenticatie in versie 3
- Priv_passphrase, voor authenticatie in versie 3
- Auth_context, voor authenticatie in versie 3

De klasse implementatie is simpelweg een wrapper voor de standaard PHP functies, waarbij er bij elke stap: get(), walk() en set() vastgelegd is wat de acceptabele variabelen zijn. Tijdens het

implementeren heeft Arno Griffioen mij de MIB databases beschikbaar gesteld voor Foundry en Brocade systemen, iets wat het identificeren van SNMP OIDs makkelijker maakt.

De SNMPCommands klasse heeft de simpele functie om de SNMP informatie per device uit de database te halen, deze als invoer te leveren aan de SNMP klasse om dat object terug te geven als bruikbaar voor de Controllers. Op deze manier overerft de SNMPCommands klasse ook de mogelijkheid SNMP queries uit te voeren.

Een voorbeeld van de implementatie hiervan is:

```
$obj = new SNMPCommands($object_id1); //Object is filled with database information from object_id1
$snmp_interface1 = $obj->doSNMP(); // Sets all SNMP variables to a new SNMP object and returns new object
$snmp_interface1->oid = '1.3.6.1.4.1.1991.1.1.2.2.1'; //Shows Version
$return = $snmp_interface1->walk(); // $return now has the output stored:
// ( SW: Version 07.8.04dT51 Copyright (c) 1996-2010 Brocade Communications Systems, Inc. )
```

7-4 PHP code SNMPobject

Effectief is het nu binnen PHP mogelijk om direct netwerk informatie op te vragen via de SNMP interface. Ik besluit nog even te wachten met het verwerken van de informatie naar bijvoorbeeld een gestructureerde Array of Object, tot ik dat ook daadwerkelijk nodig heb.

7.3.3 MySQL implementatie

Ik wil, om het MVC model effectief te kunnen toepassen ook een interface maken voor MySQL.

Nu is het keuze maken voor een MySQL implementatie op zich al een onderzoek waard, omdat er ongelofelijk veel bestaande systemen voor zijn. Tevens is er veel informatie beschikbaar omtrent het object modelleren van een type database wat daar eigenlijk niet geschikt voor is.

Feitelijk ben ik op zoek naar een ORM (Object Relational Manager). Echter op dit punt wil ik een minimale hoeveelheid tijd besteden aan onderzoek op dit gebied omdat er letterlijk duizenden ORMs op de markt zijn, die allemaal voor en nadelen hebben.

Ik besluit de standaard PEAR::DB bibliotheek te gebruiken, die standaard in het uitbreidingspakket van PHP zit, en zichzelf bewezen heeft als lichtgewicht oplossing. Deze is ook zonder veel moeite te integreren in het MVC systeem en maakt gebruik van de standaard PHP functies.

Dit systeem kent twee onderdelen:

- DataBase Connector klasse (DBC)
- DataBase Object klasse (DBO)

De DBO klasse vormt vanuit de aanroep het object, gebaseerd op de beschikbare velden. Een MySQL 'id' field word zo aanspreekbaar onder: *$dbo->id* als onderdeel van het object. De DBO klasse zorgt er ook voor dat er in de rest van de PHP code geen SQL gebruikt hoeft te worden.

De database connector gebruikt de configuratie uit Main() om verbinding te maken met de MySQL database en past filtering toe om ervoor te zorgen dat er geen problematische SQL code uitgevoerd word. Tevens zit hier een error rapportage module in. Omdat het soms niet gewenst is om directe meldingen te krijgen op het scherm, en de error rapportage verborgen zit, omdat de DBC nog voor het bestaan van de Main() applicatie al gedefinieerd word, zit er ook een mailing module in. Bij het

ontstaan van een error krijgt de developer gelijk een email met daarin de specifieke informatie zoals de SQL query, stacktrace en informatie over de gebruikte interface.

Ik kom er na tijdens het implementeren achter dat er een foute aanname gemaakt is. Ik heb de aanname gemaakt dat alle database velden over een ID veld beschikken als primaire sleutel, daar is het ORM systeem ook op gebaseerd, maar is er niet per definitie van afhankelijk.

Het ID veld is echter niet een standaard die overal doorgezet is, bij de VLAN database is bijvoorbeeld het vlan_id de primaire sleutel. Ik besluit om de afhankelijkheid van de het ID veld uit de MySQL DBO klasse te halen, en een de definitie van een primaire sleutel te accepteren. Het is vanaf dan mogelijk om in de aanroep van de DBO klasse een primaire key mee te geven.

Op dat moment is voorlopig de database interface klaar en kan er doorgegaan worden met de implementatie van een aantal Controllers en Views.

Hieronder een stuk code gebruikt om de VLAN als object uit de MySQL database te halen met DBO:

```
class Vlans extends DBO
{
    function __construct($int=NULL)
    {
        decho("Class: Constructing ".get_class($this)."() extending ".get_parent_class($this)."()..");
        parent::__construct('vlan', array('number', 'type', 'remarks', 'owner', 'contract_id') );
        if($int != NULL)
        {
            $this->load($int);
        }
    }

    function __destruct()
    {
        decho("Class: Destructing ".get_class($this)."()..");
    }
}
```

7-5 PHP code VLAN extends DBO

7.4 IST SOLL

Nu ik de interfaces voor communicatie met de databronnen heb ontwikkeld is het tijd voor de modellering. Als eerste stap voor het resultaat van de 1e cycle wil ik kunnen aantonen dat het mogelijk is om een IST/SOLL vergelijking te maken. Heel concreet wil ik:

- De status van de eerst actieve netwerk poort op een nog te definiëren switch vergelijken met de administratieve status van deze poort, zoals deze opgeslagen staat in de database. Daarbij moet, omdat deze poort online is er dus in de database staan dat deze toegewezen is een partij (klant, of eigen beheer) en deze moet op status actief staan.

Dit stelt mij op dat moment voor een probleem, wat ik voor die tijd niet heb voorzien, namelijk: de definitie van IST of SOLL hoeven niet perse eenzijdig te zijn, het is kennelijk zo dat IST niet zomaar een netwerkveld is, immers de poort moet niet alleen op de switch aanwezig zijn, maar ook op de juiste switchblade en het gaat om een eigenschap van de poort.

Aan de database is er hetzelfde probleem, de poort staat opgeslagen als entiteit, maar deze heeft allerlei informatie dat hiernaar linkt.

Op zich hoeft dit geen probleem te zijn, maar dan moet er in het ontwerp rekening mee worden gehouden dat IST en SOLL niet één specifiek dataveld is, maar een conditioneel statement.

Bij voorgaande modellen is daar geen rekening mee gehouden. De oplossing die ik wil proberen is het volgende: Voor database (SOLL) statements wil ik een aparte tabel bijhouden die naar relevante tabellen linken

Voor netwerk (IST) informatie is het nog een stapje lastiger, namelijk er zal bijgehouden moeten worden over welk netwerk device er informatie beschikbaar moet zijn, en hoe deze in verhouding staat tot de hardware. Ook moet het al vast liggen welke data query methode er gebruikt moet worden (SNMP of CLI?). Het is dus nodig voor elke individuele IST informatie aanvraag om te bepalen welke hardware dit is, en welke klasse er gebruikt moet worden, deze data zal vastgelegd moeten worden en dat is op dit moment niet zo.

Dit leid vrij direct tot de conclusie dat het ontwerp zoals dat nu beschikbaar is, misschien wel aan de eisen en wensen voldoet, maar compleet niet praktisch is. Ik besluit een gesprek met de opdrachtgever aan te gaan om hem deze informatie voor te leggen en samen tot een oplossing te komen. Zeker omdat het op dit punt al 2 weken verder is en er nog altijd geen feedback op het ontwerp is. In afwachting van het gesprek ga ik wel verder met de Views, omdat een aantal dagen wachten geen oplossing is.

7.5 Unit testing

Ik maak als laatste nog een kleine Unit-testing suitte. Dit stuk software, wat ik initieel als onderdeel van het Main() programma opneem, kan ik door middel van een configuratie alle interfaces laten testen.

Het principe is simpel, hieronder een voorbeeld, opgezet in pseudocode

functie geefRapport (id = 1) moet resulteren in een rapport object.

De unittest hierbij zal dan zijn:

functie test_geefRapport()

als geefRapport(id = 1) een object terug geeft

Unit test succesvol

Indien foutief, treed er een error op en faalt de functie.

als geefRapport() geen object terug geeft (foutive invoer)

Unit test succesvol

Indien foutief, treed er een error op en faalt de functie.

Het slagen van alle unit tests is criterium voor het beginnen van het testen op systeemeisen.

De unit tests worden vastgelegd in het configuratie bestand. De functies worden dynamisch gegenereerd.

7.6 User Interface

Nu de Modellen en Controllers enige vorm hebben gekregen is het tijd voor de Views.

7.6.1 Commandline Interface

Allereerst is er een simpele interface die dient voor de ontwikkelen, deze geeft de tekst, indien uitgevoerd vanaf de shell direct weer op het scherm. Het enige wat ik aan vormgeving gedaan heb is het ervoor zorgen dat de informatie netjes op het scherm weergeven word, en dat er bij elke debug lijn ook staat welke module voor dit bericht verantwoordelijk is.

Deze interface word automatisch gebruikt als het framework detecteert dat deze vanaf de commandline word aangeroepen. Het doel van deze interface is debugging en als tool te kunnen dienen voor de cronjob omgeving.

De commandline interface accepteert in deze cycle ook de '--debug' variabele. De bedoeling is dat er ook een systeem komt dat een lijst met variabelen die met de volgende syntax ingegeven kan worden:

'php cron.php variabele1=waarde1 variabele2=waarde2'

Deze optie word als feature in de lijst gezet voor cycle 2. Op deze manier zouden er verschillende cronjobs gecreëerd kunnen worden die ieder een aparte lijst jobs verwerken. Op die manier is het mogelijk om deze tool multithreaded uit te voeren, iets wat PHP standaard niet ondersteund. Dit is geen vereiste, maar het komt efficiency ten goed, iets wat zou kunnen zorgen voor meer draagvlak.

7.6.2 HTTP interface

Ik heb een simpele prototype ontworpen op dezelfde manier als ik de framework heb ontworpen. De webinterface in HTML word op deze manier benaderd als View. Het ontwerp gaat er van uit dat er op deze pagina een drietal kolommen komt te staan:

- Een kolom met actieve meldingen, deze zijn dus niet als gelezen aangemerkt, en zijn ten tijden van het lezen nog altijd van kracht.
- Een kolom met 'acknowledged' meldingen. Deze meldingen zijn aangemerkt als gelezen en bekend, maar er is geen intentie dat op een korte termijn op te lossen. De melding is hier nog altijd van kracht, maar er komen geen nieuwe meldingen meer voor binnen.
- De historische tabel.

De interface moet er als volgt uit te komen zien:

7-6 Concept interface

Hierbij zijn de kleuren en is de style van het huidige Intranet gebruikt.

Het idee, om de interface simpel te houden is dat er geen hinderlijke forms of grote datavelden op het scherm te plaatsen. De verwerking van events, bijvoorbeeld het acknowledgen van een melding, zal direct plaats vinden door middel van een XMLHTTP request. De melding zal daarom direct verwerkt worden. Omdat de interface objecten weergeeft en terugstuurt, kan het MVC model hier gelijk mee overweg. Het enige wat er nodig is, is een PHP interface die de JSON (JavaScript Object Notation) objecten die de interface genereert, weer terugzetten naar PHP objecten die direct weer verwerkt worden. Deze interface noem ik de Factory.

Dit bied het voordeel dat er voor alle objecten maar één object en event handler hoeft te zijn, in plaats van een handler voor elke form op de pagina.

Er zit nog een enorm voordeel aan: de webinterface is modulair, dat maakt het simpel er aan verder te ontwikkelen. Het aanpassen eraan vereist alleen een modificatie aan de View, en indien structureel een modificatie aan de Controller en het Model, maar alle veranderingen worden automatisch opgepakt door het MVC model.

7.6.3 Peer review

Ik besluit, nog voor het gesprek met de opdrachtgever, om deze interface aan de directe gebruikers voor te leggen, immers zij zullen deze moeten gaan gebruiken.

Er is direct kritiek, en het gaat om de volgende punten:

- De style komt niet 100% overeen met de rest van het intranet, want de rest van het intranet benut de pagina voor de volle breedte, waar dit ontwerp gebruik maakt van tussenruimte.
- Men ervaart deze interface als counter-intuitief omdat de rest van het intranet gebruik maakt van de forms en drop-down boxes. De gebruikte radioboxes en groep selectie systeem valt erg buiten de smaak.
- Javascript schijnt een doodzonde te zijn, want: er zijn nog mensen die gebruik maken van oudere browsers en ze zijn erg huiverig om Javascript te accepteren, dat zou het XMLHTTP ontwerp gelijk overbodig maken.
- En geheel terecht, de globale header van het Intranet ontbreekt nog in het concept.

De conclusie is eigenlijk: er is niks goed aan.

Het lijkt mij geen goed signaal. En ik besluit het gesprek met de opdrachtgever af te wachten. Ik vermoed dat het ontwerp voor de interface opnieuw moet. Ik begin mij ook te beseffen dat het ontwerp in een eerdere fase had gemoeten, zelfs in de Inception fase.

7.7 Moment van sturing

Na 2.5 week komt er het moment dat ik met de opdrachtgever in gesprek ga. We hebben het tijdens dit gesprek voornamelijk over het ontwerp, de feedback van de opdrachtgever op het Elaboration document is al vastgelegd in de opmerkingen die op het document geschreven zijn. Het gesprek wat volgt zorgt ervoor dat die notities eigenlijk niet meer relevant zijn.

De opdrachtgever merkt op dat na het zien van mijn ontwerp, hij in zag dat wat hij in gedachten heeft gehad tot die tijd, niet is wat er op papier staat. Hij geeft ook aan dat dit niet direct mijn fout hoeft te zijn, maar dat hij er iets te simpel over dacht. Als ik uitleg geef bij punten in het Elaboration document word duidelijk dat mijn ontwerp gebaseerd is op:

- Normalisatie van data.
- Normalisatie in het netwerk.
- Enkele IST/SOLL vergelijking.
- Verwerkbaarheid van de code.
- Draagvlak voor de applicatie dmv degelijke code, en daarna pas bewijzen.

Echter de opdrachtgever wil:

- Geen wijzigingen aan de huidige data(opslag).
- Meervoudige IST/SOLL vergelijkingen.
- Een situatie waarbij IST niet persé het netwerk is, maar ook de database kan zijn. Dat brengt met zich mee dat metadata over de IST/SOLL vergelijking bijgehouden zal moeten worden.
- Een prototype, niet zozeer iets moois.
- Draagvlak voor de applicatie dmv bewijzen van activiteiten.
- Simpele configuratie (het mag geen enkel beetje tijd kosten voor de engineers)

We realiseren ons dat het ontwerp zoals dat voor ons ligt, niet te realiseren valt binnen de resterende termijn van 3 weken. Omdat de opdrachtgever erg graag ziet wat er wel mogelijk is, word het doel van de software, van het werken naar productie software omgezet naar het doel van oriëntatie. De software moet de opdrachtgever inzicht gaan bieden in het IST SOLL model en moet wat de opdrachtgever betreft minimaal in staat zijn een enkelvoudige IST/SOLL vergelijking te maken.

Er speelt echter nog een groot probleem mee. Het ontwerp van het framework, dat alle oorspronkelijke eisen implementeert kent een uitgebreide configuratie. Te uitgebreid meent de opdrachtgever, omdat er per dataveld, per poort bepaald moet worden wat de testconfiguratie is. De opdrachtgever vraagt mij te onderzoeken wat de mogelijkheden zijn tot een auto-configuratie tool.

Ik geef aan, in te zien dat omdat het week 13 is, er niet genoeg tijd gaat zijn om beide te realiseren. De opdrachtgever geeft aan dat het realiseren van werkende interfaces, en de demonstratie daarvan met unit-tests voldoende uitsluitsel moet geven over het functioneren.

Het doel van de software is nu als basis te dienen voor een vervolgtraject. Dit zou iemand anders kunnen oppakken, dus het blijft zaak dat de code leesbaar is, en gedocumenteerd is.

We besluiten geen aandacht meer te bestenden aan VLANs of waves. De enig overblijvende objecten die zowel in de database staan en in het netwerk beschikbaar zijn, zijn de netwerk poorten en peerings. Het bijgestelde doel luid:

- Creëer een proof of concept van de tool die een IST/SOLL vergelijking kan aantonen over de status van een netwerk poort.

We besluiten ook dat het Elaboration document niet opnieuw opgesteld hoeft te worden, immers de nieuwe doelstellingen zijn effectief opgesteld in het bijzijn en in overeenstemming met de opdrachtgever. Wijzigingen dienen wel gedocumenteerd te worden, en er moet notitie van gemaakt worden dat het Elaboration document onder deze nieuwe voorwaarden is goedgekeurd.

Het gesprek lost nog niet alles op. Zo blijven de volgende vragen over die de opdrachtgever niet kan beantwoorden, mede omdat de informatie erg tegenspreekt:

- Hoe kan er een enkelvoudige vergelijking komen van een poort als deze niet uit één dataveld bestaat, maar uit meerdere?
- Hoe word er bepaald, zonder verder beschikbare data, wie nu de IST en wie nu de SOLL is?
- Hoe moeilijk mag de configuratie zijn, als deze geen enkele tijd mag kosten om in te stellen?

Het stelt mijzelf voor de vraag of dit het einde is van het huidige traject en er een nieuw traject moet komen. Tevens zou dat het einde van het project betekenen omdat er op dit moment nog maar 4 weken over zijn.

7.8 Procesbewaking

Aan het einde van week 13 besluit ik toch dat het de moeite waard is een poging te wagen het gewenste prototype te realiseren. Op het moment dat dit niet mocht lukken, omdat het bijvoorbeeld niet realistisch blijkt, dan zal de huidige code vrijgegeven worden aan de opdrachtgever en zal er een opleverdocument worden opgeleverd waarin de code, zoals AUP voorschrijft, worden toegelicht zonder overbodige ballast. Het zal een korte handleiding zijn tot de persoon die het project oppakt. Tevens spreek ik af met de opdrachtgever mondelinge toelichting te geven als het zover mocht komen.

Wat de kwaliteitseisen van de constructiefase betreft:

- Er is tot zover een succesvolle implementatie van de interfaces.
- Er is een succesvolle implementatie van een onderliggend framework, waarbij dit systeem te benaderen moet zijn via de webinterface en de commandline (cronjob).

Echter:

- Er is geen implementatie van een goedgekeurd ontwerp.
- De webinterface is niet als geschikt gemarkeerd, maar dat heeft nu geen prioriteit meer.

Gezien de omstandigheden, denk ik dat op dit punt dat de geschreven code, voor het grootste deel bruikbaar is, daar er nog geen uitgebreide modellen zijn gecreëerd. Dit betekend ook dat het voornemen zo'n dynamisch framework op te zetten goed uitpakt, het is in zijn volledigheid bruikbaar voor de vernieuwde implementatie.

Effectief is er met dit gesprek een einde gekomen aan de eerste spring cycle. Ik besluit door te gaan met de Construction fase, cycle 2, en cycle 1 als vruchtbare basis te beschouwen voor verder werk.

8 Construction fase - Sprint #2

Deze sprint start in week 14, met nog 1 week over voor de Construction fase in de planning. Het gehele project zal nog 3 weken duren. De laatste twee weken zijn gereserveerd voor de Transition fase, maar het word duidelijk dat deze niet meer haalbaar is. In plaats daarvan word de Construction fase met 2 weken verlengd. Deze extra tijd geeft nog ruimte voor 1 sprint cycle. De Transition fase zal simpelweg bestaan uit de oplevering van de code aan de opdrachtgever

8.1 Sprint cycle #2

Deze cycle begint direct met problemen. Er blijven nog de vragen hangen uit de 1e cycle en het begint steeds duidelijker te worden dat deze vragen ontstaan zijn uit conflicterende eisen.

Echter, druk vanuit het bedrijf om het project af te maken omdat daar een fikse beloning tegenover staat, en het prospect van een baan bij het bedrijf laten mij ertoe verleiden het project toch op te pakken.

Daarnaast is er de overtuiging dat het MVC framework dusdanig flexibel is dat essentiële delen van het ontwerp zonder problemen aangepast kunnen worden. In de praktijk blijkt dit zonder meer waar, maar dat lost de ontwerpproblemen niet op. Het toont wat mij betreft wel de kwaliteit van het product aan, en genoeg kennis van de onderliggende netwerk en database lagen om dat mogelijk te kunnen maken.

De twee weken die volgen worden gevuld met het testen van SNMP en NAL functionaliteit, het uitwerken van de unit-test suite en het herschrijven van het ontwerp door middel van testen en prototyping. De interface word gereduceerd tot een basale HTML pagina met de uitvoer van de tool, zoals deze ook op de commandline beschikbaar is. Het gaat immers nu niet meer om de interface, maar om het proof of concept.

De aanname word gemaakt dat bij het maken van een IST/SOLL vergelijking de database de IST rol neemt, en daardoor dicteert wat er in het netwerk zou moeten staan. Maar dat lost nog steeds het probleem niet op met de meervoudige IST/SOLL vergelijking.

Ik besluit om de vergelijking enkelvoudig te maken en vast te leggen dat één bepaalde poort een SNMP OID heeft en dat dit vergeleken word met een vastgelegd veld uit de database. Dit neemt voorlopig ook de moeilijkheid met de configuratie weg, immers er is voor dit proof of concept geen enkele configuratie mogelijk.

Om dit aan te tonen is er nog één conditie waar aan voldaan moet worden: er moet een live switch beschikbaar zijn die ik om informatie kan vragen, om een vergelijking te maken. Dit levert gelijk het volgende probleem op, ik mag helemaal niet met switches praten van de backbone afdeling!

De tests die ik tot dan toe gedaan heb, zijn op een switch waarvan mij bij mijn onderzoek van verteld was dat deze niet in productie was. Echter tijdens het verloop is deze switch wel actief geworden en is het niet de bedoeling dat ik er verder mee communiceer. Ik heb uiteraard wel een live switch nodig waarvan de poort als actief geregistreerd staat in de database, en besluit te overleggen met backbone. Echter, het loopt richting het einde van de week en de engineers van backbone werken veel in het veld. De daarop volgende week is week 17. En het is tijd voor een gesprek met de

opdrachtgever over de afronding van het project, omdat volgens de planning het nu tijd is voor de Transition fase.

8.2 Keuze om het project te verlengen

In dit overleg met de opdrachtgever geef ik aan dat ik heel dicht bij een werkend proof of concept ben, en dat ik met een beetje extra tijd het PoC eventueel kan aanleveren. Dit is nog steeds gebaseerd op de eerdere druk van het bedrijf, en een stukje trots. Het zit me erg dwars dat een op het eerste zicht zo simpele oplossing niet leid tot een functioneel prototype, zeker met zo'n dynamisch framework. Het helpt zeker niet dat ik veel tegengas krijg van mijn collega's die op dat moment al een aantal weken vragen wanneer 'er nu eindelijk een einde aan komt'. Dat komt een deel voort uit onbegrip voor een gepland project. Immers het lijkt binnen NLix een raar iets om met een gedocumenteerde planning te komen. Anderzijds is er onbegrip over de hoeveelheid werk gaat, in een ogenschijnlijk simpel systeem. Zoals zij eerder al vermelde 'Als er software nodig is, schrijven we het zelf wel'.

Ik besluit zelf om van de 17 weken, 20 te maken, door deze te verlengen met 3 weken. De eerste, huidige week 17, 18 en 19 zullen gevuld worden met Construction. De laatste week is voor de Transition fase gereserveerd. Het plan word kracht bij gezet door een harde einddatum in te stellen van 16 december.

In de praktijk kwam er niks van deze uitbreiding terecht. Eerst werd ik zelf ziek, en vervolgens moest ik week naar het buitenland voor een familie aangelegenheid. Dat heeft mij twee weken gekost. Het laat alleen nog maar de Transition fase over.

8.3 Procesbewaking

De Construction fase word gekenmerkt door:

- Het uitwerken van de software tot het punt waar deze klaar is voor het testen.
- Het uitwerken van de architectuur en het ontwerp door middel van sprint cycli.
- Het unit testen van de software.

Alle drie de doelen zijn slechts deels behaald. Het ontwerp, zoals deze is ontworpen vanuit de eisen en wensen is niet zoals de opdrachtgever het beoogd, en is op diens verzoek gewijzigd. Er is deze cycle wel alle moeite gedaan om tot zover te komen. Het afrondingsgesprek concludeert dat het project daarmee feitelijk ten einde is.

Verder is er aan het begin van de fase gesteld:

De Construction fase sluit af met het opleveren en de goedkeuring van de software voor productie. Tevens moet er in deze fase testrapportage gegenereerd worden om de productie geschiktheid van de software te kunnen garanderen.

Ook deze doelstelling is niet gehaald. De status van het project is daarmee: afgerond met een nieuw doel. Feitelijk is dit basis voor een geheel nieuw traject.

9 Afronding

Het enige wat nog resteert is de oplevering van het werk dat tot zover gedaan is aan de opdrachtgever met daarbij de 'release-notes' van de software. Dit laatste gebeurt per email, na een mondelinge toelichting van de resultaten tot dan toe. De opdrachtgever accepteert dat daarmee het project is afgerond.

Er vind een bespreking plaats waarin beide partijen identificeren waar de oorzaak voor het afronden van het project ligt, en het lijkt aan de volgende factoren te liggen:

- Er is onvoldoende onderzoek gedaan naar de gestelde eisen en wensen om de realiteit daarvan te kunnen beoordelen. De oorzaak ligt deels bij de conflicterende eisen en deels bij de mentaliteit van het bedrijf om geen precisie na te streven in het ontwerpen ('liever snel een prototype dan een volledig concept').
- Het ontbreken van documentatie aan de kant van de NLix, aanvulling door middel van onderzoek heeft heel veel tijd gekost.
- De opdrachtgever heeft altijd een soort Cacti of Nagios voor ogen gehad, iets wat niet realistisch bleek. Ook na gesprekken daarover is het beeld gebleven.
- Er is nooit veel ruimte geweest voor feedback. Ondanks de hoop dat de opdrachtgever daar meer tijd voor had, is in de praktijk gebleken dat deze het erg druk had.
- Er is binnen NLix nooit een degelijke basis geweest voor het projectmatig inrichten van software, er is ook geen ondersteuning geweest voor processen als planning en het modelleren.
- Ik heb zelf de fout gemaakt om een trend te hebben willen zetten met het systeem, waardoor er mogelijk signalen over het hoofd gezien zijn. Hiermee doel ik op de manier van programmeren en het inrichten van een object georiënteerde tak van het Intranet. De opdrachtgever legt uit dat het niet loont om alles om te schrijven, simpelweg omdat het teveel tijd kost en er geen backup omgeving is, indien de ontwikkelingen het intranet platleggen.

Het gesprek word afgerond en de code word op de server, gedocumenteerd, achter gelaten. Het oplever document (Hoofdstuk 12.9) word verstuurd naar de opdrachtgever. Het project is hiermee formeel afgerond.

10 Evaluatie

Voordat ik met de product en procesevaluatie begin wil ik nog een aantal punten aanbrengen, waarvan ik vind dat deze belicht moeten worden.

Keuze van bedrijf en opdracht. Ondanks dat ik gewaarschuwd was over het aannemen van een afstudeer opdracht in een klein bedrijf, vond ik bij NLix toch de kans waar ik naar op zoek was. Het was, zoals ik verwacht had een flinke uitdaging om enerzijds draagvlak te creëren voor een oplossing en anderzijds een uitdaging op het netwerk vlak om in een organisatie te stappen waar de gemiddelde medewerker 15 jaar ervaring heeft. Dat plaatst je in de situatie waar mensen er van uit gaan dat je bepaalde kennis reeds bezig. Het is dan ook zaak geweest om zo snel mogelijk zo snel mogelijk bij te scholen.

De tweede uitdaging was het creëren van een werkmethodiek in een omgeving waar iedere engineer z'n eigen methodiek toepast, waarvan geen formele documentatie is of waarvan deze niet algemeen toepasbaar is. Achteraf gezien had een opdracht met een eventueel vooropgezet framework van methodiek tot betere prestaties kunnen leveren. Structuur en orde is belangrijker voor de prestaties dan ik dacht.

Betreffende de inhoud van de opdracht, ik vond het jammer niet zo direct betrokken te zijn met het netwerk zelf. Ik heb directe toegang tot alles gehad, ter lering, maar ik heb nergens echt iets mogen toepassen totdat mijn werk zich zou bewijzen, wat zich in de ogen van de engineers niet gelukt is.

Bedrijfscultuur. Ik heb vrijwel geen probleem gehad met het passen binnen de bedrijfscultuur. Ik kon met mijn meestel collega's goed overweg en heb hele informatieve gesprekken gehad, die zowel op toepassing van de opdracht, als toegevoegde waarde hebben voor mijn verdere carrière.

Een aantal dingen hebben me toch wel tegen gehouden in het presteren. Allereerst heb ik last gehad van een collega die mij liever zag vertrekken en dat dagelijks ook bevestigde door continue in gesprekken (ook indien het deze persoon niks aanging) te bemoeien met veelal botte opmerkingen. Verder heb ik erg moeten wennen aan een nieuwe werkomgeving, waar ik niet met vertrouwde software aan de slag kon. Ook heeft het niet geholpen dat ik tot halverwege het project niet een normale zitplek heb gehad. Omdat een nieuwe fulltime collega voorgang had, heb ik heel lang op een soort van kruk moeten zitten, wat mijn rug zeker niet geholpen heeft. Ik snap wel goed dat dit te verwachten valt als stagiair.

Begeleiding. Ondanks alle goede wil heb ik het idee dat er op het gebied van begeleiding een hoop ontbrak. Er is geen structuur geweest voor vaste begeleiding en ondersteuning in de vorm van bijvoorbeeld het opstellen van documentatie en code. Ik kon op collega's direct aan, maar deze beschikte veelal niet over de kennis nodig om daadwerkelijk actief aan een software ontwikkel traject te werken. Dit heeft er zeker toe geleid dat ik bepaalde zaken gemist heb, en het uitblijven van bevestiging levert ook onzekerheid op.

Eigen iniatief. Er valt zeker iets te zeggen voor mijn eigen indicatief binnen dit project. Allereerst is er een stuk trots wat meegespeeld heeft in het project. Ik wilde graag bewijzen dat dit, in mijn ogen simpele project, ruim binnen mijn mogelijkheden ligt en ik voorzag geen problemen. Dat is een naïeve houding als ik terugkijk. Ook mijn initiatief om een technisch correcte oplossing te bieden is niet perse in goed aarde gevallen. Het leerde mij ook wel dat de technisch correcte oplossing niet de

beste hoeft te zijn! Het draagvlak creëren, maar ook het identificeren van problemen in de werkomgeving zijn erg belangrijk. Ik kan wel stellen dat deze opdracht misschien wel het verkeerde doel voor ogen had, namelijk het bewijzen van een concept, niet het creëren van een oplossing.

10.1 Producten

10.1.1 Vision document

Het visie document voldoet aan mijn gesteld eisen echter er zijn een paar verbeteringspunten te vinden:

- Het document heeft de verkeerde benaming.
- De bedrijfsprocessen hadden beter weergeven kunnen worden.
- Eigenlijk hadden er ook test-cases in dit document moeten zitten.
- Een prototype of opzet van de webinterface had hier op z'n plaats geweest.
- Agile Unified Process geïdentificeerd moeten zijn als optie, evenals beweegredenen voor de projectmethodiek.
- De organigram had beter visueel gekund en tevens kom ik er zelf niet in voor.

De kwaliteit word hier vooral gehaald uit de overzichtelijke communicatie richting de opdrachtgever. De meeste eisen en wensen zijn vastgelegd. Het enige is dat de totale acceptatie eis een 'werkende tool' is, en dat is waarschijnlijk geen goede stap geweest.

Over het geheel te zien ben ik tevreden over dit document. Is het doel van dit document bereikt? Ja.

10.1.2 Elaboration document

Het Elaboration document is naar mijn mening erg uitgebreid en bied goed inzicht in het ontwerpproces. Doordat ik UML nog niet erg effectief kon toepassen is het op UML gebied wat beperkt gebleven, maar wat er staat is mijns inziens correct. Het doel om te communiceren met de opdrachtgever is ondanks het afwijzen van het ontwerp zelf wél geslaagd, want het heeft de opdrachtgever effectief kunnen informeren van het ontwerp dat ik gemaakt heb.

Een aantal verbeteringspunten die ik zou aanbrengen:

- Duidelijker vermelden waarom RUP AUP is geworden
- UML had beter gekund achteraf gezien, maar niet op dat moment.
- De eisen zijn beter op orde dan in het Vision document, dit had grafisch beter weergeven kunnen worden.
- Er had meer tijd besteed kunnen worden aan controle op spellingfouten.

Ik ben tevreden met de kwaliteit van het document. Het voldoet mijns inziens ook aan de gestelde eisen. Ik vind het jammer dat het alleen maar goedgekeurd is vanwege het aangepaste ontwerp. Ik had graag gezien dat het feitelijke ontwerp realistisch was. Het doel heeft dit product zeker bereikt.

10.1.3 Code en opleverdocumentatie

Dit is een lastige evaluatie. De code heeft wat mij betreft de beoogde kwaliteit. Ik heb een compleet framework gecreëerd vanaf zo goed als niks, en heb dat effectief kunnen toepassen op netwerk en database communicatie. Het is goed gedocumenteerd en kan snel worden toegepast als iemand anders het project op pakt. Anderzijds is de code geen product op zichzelf, dat is namelijk het opleverdocument.

Er kleeft ook een keerzijde aan de code. Het is nog maar de vraag of de keuze deze radicaal andere stijl van code te willen implementeren ook goed gedaan heeft aan het draagvlak van de opdracht. Het roept weerstand op omdat men gewend is anders te werken. Het is ook enigszins bedreigend dat een student 'even' komt laten zien hoe het in zijn ogen wel moet.

Het IST/SOLL model kan met deze software bewezen worden , en daarom word effectief het doel uit het Vision document gehaald:

Binnen de gezette termijn van 17 weken moet inconsistentie in het IST SOLL model aan te tonen zijn door middel van een software tool dat rekening houdt met informatie uit de MySQL database en de live omstandigheden in het actieve backbone netwerk.

Effectief is er ook 17 weken aan gewerkt, en het IST/SOLL model is bewezen, zij het dat dit alleen mogelijk was met een enkelvoudige vergelijking en alleen onder vooraf opgestelde condities.

Het voornemen van de opdrachtgever om de code later in te gaan zetten is bevestiging van deze conclusie.

Het opleverdocument is zoals AUP voorschrijft kort en bondig en bied houvast voor de persoon die het project hierna oppakt. Het voldoet daarom.

10.2 Procesverslag

10-1 Leermoment (Stages of procrastination)

Het leermoment is: verbreek ten alle tijden deze vicieuze cirkel

Over de kwaliteit zelf ben ik matig tevreden. Ik heb het gevoel dat ik er meer uit had kunnen halen, maar door mijzelf de overschatten, of de hoeveelheid werk te onderschatten heb ik toch op bepaalde zaken een compromis moeten maken. Zo had ik liever NLix actief betrokken bij het opstellen ervan en had ik meer tijd willen hebben om het ontwerp, naast de ontwerpoverwegingen, meer toe te lichten. Het voldoet aan de gestelde eisen, maar niet aan mijn verwachtingen.

10.3 Projectmanagement

Het projectmanagement is een erg leervol proces geweest. Allereerst ben ik blij dat ik na eerst RUP toch gekozen heb voor AUP. AUP bood mij net even de extra flexibiliteit die ik nodig had in een organisatie waar de structuur voor een dergelijk project ontbreekt. Als ik terug kijk heb ik AUP niet 100% tot op de letter toegepast, maar dat strookt met de Unified Process gedachte, waarbij het proces ondersteunend is aan de opdracht, niet leidend.

Een aantal zaken die ik anders gedaan zou hebben:

- De communicatie met de opdrachtgever beter vastleggen.
- Meer tijd nemen voor feedbackmomenten.
- Meer peer review en ondersteuning zoeken op het gebied van planning, risico's en haalbaarheid.

Ik zou AUP met plezier nogmaals inzetten. De kracht zit hem in de Inception fase, die ik helaas niet helemaal heb kunnen benutten ivm de overstap van RUP. Ik zou het in de toekomst ook iets agressiever inzetten, zeker bij de beslissingsmoment had er meer uit gehaald kunnen worden door bijvoorbeeld een naar AUP gestylde gespreksagenda op te stellen voorgaand aan het overleg.

Is het project management effectief en succesvol geweest? Deels:

- **Modellering**. Is effectief geweest, maar niet geheel naar wens omdat ik mijn UML kennis onderschat heb, ik had mogelijk voor een andere techniek kunnen kiezen.
- **Implementatie**: Het implementatiebeheer dat de lijn: ontwerp -> implementatie volgt effectief in kunnen zetten. Ik heb het gevoel dat dit proces goed verlopen is.
- **Test:** Ik heb dit proces slechts gedeeltelijk in kunnen zetten. Ik kan wel zeggen dat het niet opnemen van de test-cases in de Inception fase een gemis is. De effectieve unit-testen in de construction fase zijn daarintegen wel duidelijk aanwezig en dienen hun doel.
- **Deployment**: Van deployment is in zo verre sprake geweest dat ik effectief een deployment script heb ontwikkeld en dat de code daadwerkelijk op een live omgeving draait. Tevens met de bijgestelde doelen is het succesvol omdat ik de code heb kunnen opleveren in een staat waarbij iemand anders deze kan overnemen.
- **Configuration management**: Ik heb de beschikbare middelen effectief in kunnen zetten. Ook heb ik degelijke afwegingen kunnen maken in ontwikkelmethodiek en de gebruikte software.
- **Project management**: De fasering was succesvol, en heb ik effectief in kunnen zetten. Ondanks dat deze niet afgerond is, immers het project mag en kan ook eerder ten einde komen. De planning had beter gekund. Ik ben tevreden met de keuze van de GANTT kaart door deze effectief als middel in te zetten. Echter had een digitale versie, die direct update misschien meer effect gehad.
- **Environment**: Ik heb gedurende het project, maar specifiek tijdens het sturingsmoment in de Construction fase effectief de omgeving van het project beheerd. De afwegingen voor het gebruik van de software pakketten zijn duidelijk gemotiveerd en gedocumenteerd.

Effectief ben ik er best tevreden mee, helemaal gezien de omgeving waarin dit project gedraaid heeft. Ik heb er een hoop van geleerd.

11 Beroepstaken

Verantwoording, per sectie, zoals aangegeven in het competentie model.

A1. Analyseren van het probleemdomein

Het onderzoeken en beschrijven van het gegeven probleemdomein (van de opdrachtgever) in termen van het TI domein, en het vaststellen van de gewenste veranderingen. Het inwerken in het domein van de opdrachtgever, maakt onderdeel van deze taak uit.

Ik heb van interviews, met als controle mechanisme peer reviews, gebruik gemaakt om een formele beschrijving van het probleemdomein te formuleren. Deze formulatie is zowel in het Vision document als in het Elaboration document terug te vinden. Ik heb deze formulering door UML modellering toe te passen effectief met de opdrachtgever kunnen communiceren. Zowel de opdrachtgever als ikzelf ben er van overtuigd dat dit een accurate weergave is van de gestelde een formele beschrijving gemaakt van probleemdomein.

Ik heb deze taak zelfstandig uitgevoerd, waarbij er licht sturende elementen aanwezig zijn, door bijvoorbeeld bevindingen die buiten de scope van de opdracht toch te adviseren aan de opdrachtgever. Tevens heb ik een actieve rol genomen bij het formuleren van de opdracht.

De opdracht heeft te maken gehad met conflicterende eisen en aanzienlijke tijddruk, getuige ook de opdrachtgever die aangeeft het project overschat te hebben. Veder ligt het project intern gevoelig omdat collega's zich bedreigd voelen omdat zij de software als een 'gimmick ' van het management zien, en niet als oplossing voor een probleem waar zij zich mee identificeren. De impact van de opdracht zelf is minimaal, het is feitelijk een test-case met als doel een prototype, maar het eiste van mij wel degelijk mijzelf verder te scholen. Ik heb vooral op het BGP gebied enorm veel geleerd. Dit doet mij ertoe schatten dat ik deze taak op een complex niveau heb uitgevoerd. Ik durf met zekerheid te zeggen dat deze taak op **niveau 4** is uitgevoerd, en behaald.

A3. Achterhalen van behoeften van belanghebbenden

Het document dat de opsomming van behoeften is samen gevat in de definitiestudie van de software dat ontworpen moet worden. Het moet duidelijk zijn dat er overeenstemming is bereikt tussen de belanghebbenden dat alle behoeften behartigd worden.

Ook bij deze beroepstaak heb ik door middel van interviews, ondersteund door peer reviews, een beeld gemaakt van de behoeften van de dit project. Ik heb geïdentificeerd welke mensen er de doelgroep zijn, en wat deze mensen wel en niet als een probleem ervaren. Ik heb hiervoor niet alleen de documentatie bekeken (zoals dat beschikbaar is op het Intranet) maar ik heb ook met een demo aan kunnen tonen wat in dit geval niet de voorkeur heeft. Ik ben niet 100% tevreden met het feit dat de test-case klant nooit verwerkt is, in dat geval had ik zeker meer succes gehad. Ik kan daarom ook niet met 100% zekerheid zeggen dat alle behoefte volledig in kaart zijn gebracht. Wel zijn de juiste technieken daarop toegepast.

Ik heb deze taak op een lastig niveau uitgevoerd op een zelfstandige manier. De argumentatie dat dit een complexe situatie is, is omdat de opdrachtgever en de doelgroep niet in overeenstemming zijn over hun belangen. De engineers hebben zeker minder belang bij de oplossing, daar het vooral van financiële aart is. Tevens is het bedreigend omdat het een controletaak voert over hun werk. Vanuit

de opdrachtgever gezien is dit juist een experiment om het model te bewijzen en heeft het belangrijke financiële motivaties. Het resultaat van deze beroepstaak is te zien in het Vision document en het Elaboration document. Ik kan concluderen dat deze beroepstaak op **niveau 3** is uitgevoerd en dat deze niet volledig succes heeft omdat er geen volledige overeenstemming is bereikt met de belanghebbenden over hun specifieke behoeften.

C8. Ontwerpen van een technisch informatie systeem

Door het uitwerken en opleveren van het systeem ontwerp moet er aangetoond worden dat er gehouden is aan de bedrijfsstandaard voor het documenteren van een systeem.

Voor het uitwerken van deze beroepstaak heb ik een technisch informatie systeem formeel op papier gezet in zowel het Elaboration document als in het opleverdocument. Ik heb daarvoor in de eerste plaats UML gebruikt, wat ik naar weten correct heb toegepast. Het bevat tevens een modelweergave van het ontwerp, waarbij de huidige database is meegenomen.

Ik heb deze taak sturend uitgevoerd, want er is standaard geen documentatie of ontwikkelstandaard binnen het bedrijf aanwezig. Door een voorbeeld te zetten, en de techniek uit te werken voor een volgend project word er aangegeven dat het ontwerp succesvol is. De software zelf mag misschien een klein doel bereiken, de opgezette framework zou ook bij andere beoogde applicaties stand houden. Ik heb de specificaties van het ontwerp direct van de eisen en wensen afgeleid. Het enige waar er bij deze beroepstaak minder op gelet is, is het passen binnen de ontwikkelstrategie. Het argument om object georiënteerde code toe te passen bijvoorbeeld is gebaseerd op technische juistheid en het feit dat een deel van de recente Intranet codebase wel op deze manier ontwikkeld is. Het is in mijn ogen vooruitgang, ook als mensen daar huiverig voor zijn. Verder is de code met voldoende documentatie en commentaar opgeleverd. Genoeg in ieder geval om iemand met enig ervaring met object georiënteerd programmeren in te zetten.

Daar het ontwerp van het systeem niet veel nieuws voor mij was, en het niet direct impact gehad heeft op het bedrijf hecht ik minder complexiteit waarde aan deze beroepstaak. De uitdaging lag binnen dit project voornamelijk op het draagvlak, meer dan op de complexiteit van dit ontworpen systeem. Ook de toegevoegde waarde van het netwerk gebied, dus communicatie met en het schrijven van bibliotheken voor netwerkapparatuur. Deze beroepstaak is lastig geweest, en sturend uitgevoerd, wat dit een beroepstaak op **niveau 4** vormt.

D17. Testen van software systemen

Het toetsen of het software systeem voldoet aan de gestelde eisen door het opstellen van een testplan en het toepassen van methoden en technieken voor het testen.

Daar het testplan in zijn geheel ontbreekt, kan er niet aan deze beroepstaak op voldoende niveau voldaan worden. Deze beroepstaak is dus niet zoals bedoeld uitgewerkt.

H6. Professioneel werken: Resultaat gericht werken

Er word door middel van het opleveren van een gestructureerd Plan van Aanpak, meer de focus op een geprioriteerde programma van eisen, aangetoond dat de opdacht zal voldoen aan de eisen en de bijbehorende kwaliteitsnormen.

Deze beroepstaak heb ik geleid uitgevoerd doordat ik mij heb laten verleiden deze taak bij te laten sturen door de opdrachtgever, ook al was dit niet gewenst in de situatie. Er is resultaat gericht gewerkt door een degelijk plan op te zetten, en deze tot op grote hoogte ook te volgen. Waar dit afweek, door bijvoorbeeld late feedback, is er actief op ingespeeld en er is actief actie ondernomen indien een dergelijk feedback moment achter bleef. Echter, het uitvoeren van deze beroepstaak tegen het einde van het project is niet geheel professioneel geweest door te makkelijk in te gaan op de wensen van de opdrachtgever en ruimte te geven voor feature creep.

Ik kan met het Vision document aantonen dat er kwaliteitseisen zijn, waar de ingeleverde documenten zich aan voldoen, en heb verantwoording aan de opdrachtgever afgelegd waar dit niet zo was. Ik heb ook aangetoond, met de keuze de Transition fase te schrappen dat er besef is van tijdsdruk. Ik heb ook ten alle tijden, met name door middel van de GANTT kaart het proces bewaakt, tevens door dit extra aandacht tijdens de Project Management dicispline van de Unified Proces methodiek te geven. Ik kan met zekerheid zeggen dat dit een lastig proces is geweest en dat maakt dit een succesvol uitgevoerde **niveau 2** beroepstaak.

In het kort:

Beroepstaak	Complexiteit	Taakrol	Niveau
A1.	Zelfstandig	Complex	4
A3.	Zelfstandig	Lastig	3
C8.	Sturend	Lastig	4
D17.	-	-	-
H6.	Geleid	Lastig	2

12 Bijlagen

12.1 Verklarende woordenlijst

API	Application Programmable Interface, een programmeerbare interface waarbij systemen zonder tussenkomst van mensen met elkaar kunnen communiceren
AS objecten	Autonomous Systems, is een aanduiding voor een netwerk met een eigen routingspolicy, dat onderhouden word door één administratieve entiteit.
AUP	Agile Unified Process, een projectbeheer methodiek
Bash	Bournce Again Shell, een veelgebruikte commandline omgeving voor Linux en POSIX compliant systemen
BGP	Border Gateway Protocol, ook wel het internet protocol, dit super robuste routing protocol wisselt routing tabellen uit
Brocade	Brocade Communication Systems Inc. Een leverancier van netwerksystemen voor LAN-, MAN- en WAN netwerken. Oorspronkelijk SAN leverancier
Cacti	Een monitoring tool die gebaseerd is op PHP, MySQL en RRDTool http://cacti.net/
CGI	Common Gateway Interface, een gateway voor het dynamisch bewerken van uitgaande HTML pagina's op webservers
CLI	CommandLine Interface, verzamelwoord voor een terminal of invoersysteem voor de gebruiker
Daemon	Daemon, een service systeemprocess wat alleen reageert of binnenkomende verzoeken
DBC	DataBase Connector, zoals gebruikt in dit project
DBO	DataBase Object, zoals gebruikt in dit project
Enabled (mode)	Een modus waarin een router of switch bewerkt kan worden door een beheerder, voornamelijk in een Cisco achtige omgeving.
EUP	Enterprise Unified Process, de Enterprise versie van RUP, ontwikkeld door IBM
Exception Handler	Een event handler die specifiek errors afhandeld
False-positives	Meldingen die een fout weergeven, terwijl het eigenlijk geen fout is
Foundry	Voormalig leverancier van LAN-, MAN- en WAN netwerkapparatuur, opgekocht door Brocade
IDE	Intregraded Development Environment, software dat de ontwikkeling van nieuwe software ondersteund
INNODB	Een bekend type MySQL database wat een relationele Foreign Key en database transacties ondersteund
Internet Exchange	Een platform waarop AS netwerken met elkaar verkeer en (BGP) routing informatie uitdelen
IST/SOLL	Duits voor: Is en zal zijn. Een model om de waarheid met een maatstaaf te controleren
JSON	JavaScript Object Notation, een industrie standaard voor de notitie van gestructureerde objecten in een browser omgeving
MIB	Management Information Base, een SNMP database met object meta informatie
MRV	MRV Communications, een leverancier van geintregreerde circuits en multiplexers
Multiplexer	Ook wel Mux, een apparaat wat meerdere signalen over een enkele lijn propageert
MVC	Model, View and Controller. Een modellering om binnen een applicatie scheidingen te kunnen leggen tussen de interfaces, de controller en de gebruiker.
MyISAM	Een bekend type MySQL database, redelijk verouderd, heeft weinig ondersteuning voor relationeel ontwerp en database normalisatie
MySQL	Een Open Source implementatie van de SQL server standaard. Word momenteel door Oracle beheerd
Nagios	Nagios is een bewezen, robuust stuk networking monitoring software. Vormt zo'n beetje de

	industrie standaard voor systeem administratie
NAL	Network Abstraction Layer, geschreven door Vincent Bourgonjen
Nlix	Netherlands Internet Exchange, opdracht gever en gastbedrijf binnen dit project
NOC	Network Operations Center, een typische 'war-room' van het bedrijf dat infrastructuur beheert
OID	Object Identifier, een SNMP benaming voor een object pointer
PAL For NAL	De implementatie voor NAL, zoals gebruikt in het project
Peering group	Een group mensen die besluit met elkaar informatie uit te wisellen. Binnen BGP ook als datatype beschikbaar
PHP5 OO	Pre-Hypertext Processor, versie 5, Object geOrrienteerde notatie
PHPMyAdmin	Een populair PHP gebaseerd framework on MySQL servers te beheren
Route server	Een BGP router die geen verkeer accepteert, maar slechts binnen een internet exchange routes uitdeelt.
RUP	Rational Unified Process, een projectbeheer methodiek, ingevoerd door IBM
SCP	Secure CoPy, een SSH methode om (beveiligd) data te transporteren van één machine naar een ander
shell	Shell is een algemene indicatie voor een CLI omgeving
SNMP	Simple Network Management Protocol, een industrie standaard voor het beheren van informatie stromen van een naar beheerde hardware
SQL	Structured Query Language is een taal om te communiceren met relationele database management systemen
SSH	Secure Shell, een shell toegankelijk over TCP/IP. Een veelgebruikte tool voor Linux systeembeheer. Ondersteund SCP
syntax-highlighting	Een techniek waarbij een IDE code patronen verschillende kleuren geeft om makkelijk visueel onderscheid te kunnen maken
Syslog	Een industriele standaard voor datalogging. Het is een simpel bericht systeem waarbij clients hun logbestanden naar een centrale server sturen
Telnet	TELetype NETwork is, net als SSH, een applicatie om over TCP/IP een shell te benaderen. Telnet kent geen beveiliging
UML	Unified Modelling Language, een zogenaamde markup language om object georrienteerde informatie systematisch weer te geven
Unit-test	Zie hoofdstuk 7.5
VLAN	Virtual LAN, een techniek om op een switched platform verkeer van elkaar te scheiden op broadcast domein
XMLHTTP	Een Javascript API om XML en JSON te communiceren over HTTP requests

12.2 Bronvermelding

(sd). Opgeroepen op 02 01, 2012, van NLix website: http://www.nl-ix.net/about/company/

Ambler, S. (2009, 01 01). *The Agile Unified Process*. Opgeroepen op 02 15, 2012, van Ambysoft.com: http://www.ambysoft.com/unifiedprocess/agileUP.html

Ambler, S. W. (2005, 01 01). *incrementalReleases.* Opgeroepen op 02 16, 2012, van Ambysoft: http://www.ambysoft.com/artwork/incrementalReleases.gif

Ambler, S. W. (2005, 01 01). *lifecycleAgileUP.* Opgeroepen op 02 15, 2012, van Ambysoft: http://www.ambysoft.com/artwork/lifecycleAgileUP.gif

Ambler, S. W. (2005). *ManagersIntroToRUP.pdf.* Opgeroepen op 2011, van Amysoft: http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

Gansner, & North. (2011). *Dot Language.* Opgeroepen op 2011, van Graphviz.org: http://www.graphviz.org/content/dot-language

Gansner, E., & North, S. (2012). *Graphviz - Credits.* Opgeroepen op 2011, van Graphviz.org: http://www.graphviz.org/Credits.php

Hoogendoorn, S. (2011). *Pragmatisch modelleren met UML 2.0.* Amsterdam: Pearson Education Benelux.

NLix statistieken. (sd). Opgeroepen op 02 02, 2012, van NLix - Netherlands Internet Exchange: http://www.nl-ix.net/network/statistics/

PHP.net. (2012). *history.php.php.* Opgeroepen op 2012, van PHP.net: http://nl3.php.net/manual/en/history.php.php

RUP Fundamentals. (sd). Opgeroepen op 2011, van NiH eRA: http://era.nih.gov/docs/rup_fundamentals.htm

Stages of procrastination, J. (sd). Opgehaald van Tumblr: http://chibird.tumblr.com/

Usenet op Wikipedia. (sd). Opgeroepen op 02 02, 2012, van Wikipedia.org: http://nl.wikipedia.org/wiki/Usenet

van Beijnum, I. (sd). Opgeroepen op 02 05, 2012, van BGPexpert.com: http://www.bgpexpert.com/books.php

12.3 Figuren

12.4 Tabellen

12.5 Gespreksverslagen

Gesprekverslag 31-05-2011

Genodigden: Jan Hogeboom
Datum: 31 mei 2011 – 10:00
Onderwerp: Introductie gesprek

Onderstaand is een lijst met gesprekspunten en bijbehorende notities:

- Het plan word voorgelegd aan Jan ter inzage, en op een paar kleine textuele fouten is alles in orde. De notitie is dat er in het Plan van Aanpak gewerkt moet worden aan een duidelijke planning.
- De uitloop ondersteunen tot 23 december is geen probleem. Het gaat dan vooral om revisies, contact per mail en telefoon. Er word duidelijk gemaakt dat het uitloop traject het resultaat is van eerder beginnen aan het afstudeertraject.
- Praktische zaken worden besproken
    - Werktijden -> 9 tot 5 met de mogelijkheid de uren zelf aan te passen, zolang er maar 8 uur per dag gewerkt word. Vertraging / afwezigheid zelf compenseren met verdure uren en dagen. Er zijn geen vakantiedagen anders dan de nationale feestdagen.
    - Werkstation -> Is pas beschikbaar tot volgende woensdag, tot die tijd word er gebruik gemaakt van een tijdelijke oplossing (noot: is een eigen laptop, zoals later per mail besproken). Het workstation zal Linux Debian zijn en vereist naast een tekstverwerker verder geen specifieke software. Er zal vooral met PHP en MySQL op gewerkt worden, mede als SSH en terminal achtige verbindingen.
    - Er word geen gebruik gemaakt van software die specifieke licenties of verdere training vereist.
    - Begeleiding word op een vast moment ingepland. Vanwege de dynamische agenda van de begeleider (Jan zelf) zal hier flexibel mee worden omgegaan. Er word duidelijk gemaakt dat dit geen enorme druk op de agenda hoeft te zijn.

12.5.1 Email naar Sales

Managed peering

Managed wave for fibrechannel storage buiten routing om
Gepatched op de kast in het DC

2 managed routers op 2 locaties (Databarn - Schuberg Philis)

Beste,

Mijn naam is Robert Jansen en ik werk als beheerder voor NetVPS. We zijn een virtuele server provider en we willen graag af van de afhankelijkheid van onze huidige netwerkaanbieder (Databarn/wedare) en we willen de stap aan gaan om het eigen netwerk te routeren. Tevens willen we ons netwerk IPv6 ready maken.

Nu heb ik begrepen van conculegas dat NL-IX mij kan helpen bij het opzetten en inrichten van de nieuwe situatie. We willen graag op twee locaties actief worden, namelijk in het oude datacentrum Databarn en een tweetal nieuwe racks in Telecity 4. We hebben echter niet de kennis in huis om de routers te beheren en netwerkverbindingen op te zetten. Hoe dan ook, we willen onze eigen IP adressen zodat we niet afhankelijk zijn van het datacenter bij een verhuizing.

Verder kwam ik op de website tegen dat jullie ook point to point verbindingen aanbieden tussen datacentra. Nu hebben we straks, om de servercloud geheel redundant te maken ook SAN replicatie nodig en we willen niet gebruik maken onze metered verbinding die we zoals nu per Mbit betalen, wat kunnen jullie voor ons betekenen?

Ik zou antwoord per email erg waarderen, daar ik telefonisch slecht bereikbaar ben de komende tijd.

Met vriendelijke groet,

12.5.2 Gespreksverslag 13-07-11

Voortganggesprek Jan 13 jul

Besproken:
Huidige codebase / new framework?
Huidige codebase gebruiken voor portabiliteit en onderhoudbaarhied na vertrek
Ontwerp model kort toegelicht
Er word toegelicht hoe de opdracht zich moet beperken bij netwerk synchronisatie en dat database / database synchronisatie geheel buiten de scope valt
Verduidelijking database diagram
Policy (zoals naamgeving) bespreken met techniek alvorens dit voor te stellen in elaboration document.
DB visualisatie werkt niet (onpraktisch), geeft wel weer dat de database gesnapt is en dat het model digitaal verwerkt is, wat dan weer wel praktisch is tijdens het verwerken.
Voortgang besproken
Vrijdag draft document
Bevat minimaal eisen, wensen en policy voorstel
Voorstel gedaan nieuwe planning (ivm verschuiving a 3 weken totaal)
Ingepland in Agenda, data per email

12.5.3 Gespreksverslag

Gespreksverslag 28-06-2011

Aanwezigen: Jan Hoogenboom, Gerben Geijteman

Onderwerp: Voortgang elaboratie fase en goedkeuring Initiation document.

Betreffende het Initiation document:
In overeenstemming is er besloten om na de vierde week in de elaboratie fase een draft document in te dienen om zo meer ruimte te geven voor sturing (vrijdag 10 juni).
Er word uitgeweid over de invulling van de twee construction cycli. Namelijk, een cycle zal gebruikt worden voor core functionaliteit van de tool, het synchronisatie proces en de data miners. De tweede cycle is voornamelijk gefocust op de user interface, waarbij het belangerijk is dat de tool ook bruikbaar is, inclusief de wens voor een 'acknowledge' en melding functionaliteit.
Goedkeuring word verleend om verder te gaan met het proces.

Betreffende de elaboratie fase:
De beschrijving van catagorieen groeperen in onderwerpen zoals poorten, vlans en links.
Jan duid aan dat het netwerk ook leidend kan zijn bij het samenvatten van relevante datavelden.
Dit kan er toe leiden dat informatie die verzameld word (nog) niet beschikbaar is in de database en dat kan synchronisatie bemoeilijken, maar bijvoorbeeld het debuggen makkelijker maken.
De beste methode om volledigheid te bepalen van de beschrijving van beschikbare en bruikbare informatie is door het opstellen van een zogenaamde uitputbare lijst. Deze dient in overeenstemming met de engineers worden opgesteld, waarbij bepaald gaat worden wat:
De impact van het verkrijgen van de informatie is op de netwerkapparatuur
De bruikbaarheid van de informatie is bij de werkzaamheden.
Waar de informatie op te halen is (en hoe).
Wat het risico is van het niet beschikbaar hebben van de informatie, bepaald door de impact dat het onbreken van deze informatie kan hebben op klanten. En daarmee is ook, hoewel niet expliciet genoemd, ook het financiele risico vastgelegd.
Na het vaststellen van een lijst met datavelden kan in overleg worden bepaald welke informatie er gebruikt gaat worden in het opstellen van de oplossing en de bijbehorende datamodellen.

Tevens word het aanstaande bedrijfsbezoek kort besproken, en dat levert geen inhoudelijke verassingen op.

12.6 Vision document

[Zie Document Bijlagen - Vision document]

12.7 Elaboration document

[Zie Document Bijlagen - Elaboration document]

12.8 GANTT kaart

[Zie Document Bijlagen - GANTT kaart]

12.9 Oplever document

Synctool - door Gerben Geijteman [gerben@hyperized.net] 15-12-2011

=> Doel - Het detecteren en rapporteren van inconsistency tussen de netwerkomgeving en de database, gefocust op interfaces (ports), BGP peers en VLANs.

=> Opzet - Een crongebaseerde tool die per tijdsinterval een check doet en dit eenmalig via email rapporteert, en tevens een rapportage op de webinterface plaatst. Het moet simpel zijn om de configuratie van de checks aan te passen, en daarvoor moet het instellen via de reeds bestaande Hardware interface van het intranet kunnen. Het huidige intranet is gebaseerd op een mix tussen procudereel en lineare programatuur. Om compatibaliteits problemen te voorkomen was de opzet met deze tool om de database handler te herschrijven voor object georrienteerde programmatuur met oog op de toekomst. De huidige PHP omgeving op de server met override derrives hebben ervoor gezorgd dat bij het huidige Intranet de error reporting uitgeschakeld staat. Dat heeft ervoor gezorgd dat deze tool ook een eigen error handler heeft. Het resultaat is los op het MVC model gebaseerd, mede door de inzet van een aparte database handler en meerdere interfaces (views). Tevens is de tool via dezelfde functies ook bereikbaar via de webinterface.

=> MVC structuur, gezien vanuit de gebruiker

```
index.php (View)
  |_ config.php (Configuration)
  |_ Main() class/main.class.php (Controller)
                |_  DBO() - class/dbo.class.php (Abstract Model)
                        |_ cli() - class/cli.class.php (Model)
```

=> Autoloader (PHP5 Magic) - config.php

Als autoloader worden er SPL (Standard PHP Library) functies gebruikt. Er is gekozen voor SPL (spl_autoload_register) boven __autoload() omdat SPL aangeraden word boven de standaard autoload. Tevens werkt __autoload() niet onder de CLI omgeving. De autoloader laad alle klasses die er nog niet ge-include zijn automatisch uit de /class directory. Vereiste is wel dat er een bestand in deze directory staat met de volgende syntax: klasse.class.php en in die file moet minimaal de klasse definitie staan. Zo niet volgt er een Exception en zal het uitvoeren van de functies die van deze klasse afhangen falen. PHP Autoload laad bestanden en klasses niet in voordat deze daadwerkelijk nodig zijn, en zorgt ervoor dat met minale code (namelijk alleen de aanroep $c = new Class()) alles word ge-include en aangeroepen.

=> Model

DBC (DataBase Connector - class/dbc.class.php) en DBO (DataBase Object - class/dbo.class.php), deze twee klasses zijn, net als Main, een uitzondering op de regel. DBO is een abstracte class die de parent vormt voor alle te includen subklasses. DBO gebruikt DBC als interface / connector naar de database. Kort gezegt bevat DBO alle object definities en functies die relateren aan object modificatie en DBC bevat alle (My)SQL gerelateerde functies.

Door de DBO/DBC opzet kan er met minimale moeite een nieuwe klasse gecreerd worden die een tabel in de database representeerd. Het enige wat er aangegeven moet worden is welke velden er beschikbaar zijn en eventuele klasse specifieke functies. Een template voor nieuwe classes is te vinden in: class/template.class.php .

Een SQL file van de database zoals deze gebruikt is tijdens de ontwikkeling staat in structuur.sql

=> View

Interfaces, roepen Main() aan en vormen een visuele shell

- index.php Webinterface, http://intranet.nl-ix.net/synctool/index.php accepteert ?debug=1 voor debugging
- cron.php Crontab script, aan te roepen via CLI of Crontab, accepteert ook --debug voor debugging
- pal.php PHP for NAL, test tool en visuele shell voor NAL, http://intranet.nl-ix.net/synctool/pal.php

=> Controller

Main(), is het hoofdprogramma. Deze klasse importeert de config en hier zit de meeste logica.

Config heeft vier functies:

1) Alle configureerbare waardes in een enkel object verzamelen
2) Error handling
3) Autoloader
4) Initieer database connector (DBO) als globaal object

Config op zijn beurt laad debugging.php, een set losse debugging functies

- class/main.class.php Hoofd klasse
- debugging.php
- config.php

=> Overige scripts

Er zijn nog een aantal scripts aanwezig die voor verdere ontwikkeling nuttig kunnen zijn:

- indexold.php Bevat een voorgesteld, maar afgekeurd ontwerp
- factory.php Bevat een factory, handig bij toepassing van asynchrone (JS/XMLHTTP) scripts op een interface
- html.html Bevat code snippets die bij het testen gebruikt zijn
- interface.php Bevat test code voor pal.php
- morse.php Bevat test code voor interfacing
- snmp.php Bevat test code voor SNMP querying met class/snmp.class.php

12.10 Uitwerking logboek, voorbeeld

Digitale notities afstuderen bij NL-IX 2011

Uitleg
- Proces is gemarkeerd met Geel
- Keuzemoment Oranje
- Aanname Groen
- Overig belangerijk met Blauw

15-06

- Bezoek aan datacentrum
- EvoSwitch en TC2 om inzicht te krijgen in de gebruikte hardware.
- MRV uitgelegd gekregen
- Dark fiber systeem uitgelegd gekregen
- Router / Switch systeem uitgelegd gekregen

16-06
- Verslag in het weekend
- Specificaties opgesteld voor een sales order, ter observatie van het afhandelen van een klant, ter doel om een gemiddelde klantcase te creeren waar later mee getest kan worden.
- Planning opgesteld voor initialisatie
- Voorlopig DB onderzoek
- BGP training
- Ipv6 training
- Bevinding: De focus van de consistency testen moeten vooral bij OpenPeering liggen.
- Identificatie van database identifiers: Poorten, interfaces, switch fabrics, peerings, link(states), IP adressen en BGP informatie.

17-06
- LEAF (switch informatie) gebruikt om meer inzicht te krijgen in organisatie van het NL-IX netwerk.
- Verslag skelet opgesteld, nav eerdere ervaring en referentie verslagen
- Overweging: Wel of geen code opleveren? Wel het ontwerp, geen code omdat de beoordeling op het ontwerp gevoerd kan worden
- NOC handboek ontdekt, bied referentie informatie, maar is niet volledig up to date, dus moet met een korreltje zout genomen worden.
- Toevoeging identificatie velden: IP allocatie, hardware profielen, klantbenamingen subnetten en VLANs
- Interview gehouden met sales (vinnie)
- Mogelijke moeilijkheden met:
- Overnames (hoe om te gaan met hernoemen)
- Entiteites mutaties
- Voor referentie zie interview verslag
- Test klantprofiel aangemaakt

20-06
- BGP training, geleerd:
- Explict deny routes
- /24 is het kleinste normaal geaccepteerde block PI space
- Filters op outgoing routes
- Routers kunnen geDDoS'ed worden door MD5 hash spoof
- 32Bit AS (paden) zijn nog moeilijk
- Naar school geweest voor feedback
- Project naar AUP ipv losse onderdelen

- Niet meer: onderzoek en uitvoeren, is intregraal een enkel project
- PVA herschrijven naar doel
- Keuze voor opstel PVA
- Tabellen maken van keuzes en afwegingen
- Criteria verantwoorden
- Het inzicht is gecreerd om proces anders in te richten
- Inception fase heeft Invision document
- Per mail tussen assesment afspreken
- Besproken vakantie dagen op te nemen om project wat te vertragen om op die manier beter gebruik te kunnen maken van feedback momenten school.

21-06
- Ipv6 cursus

22-06
- PVA herschreven
- Verder in database structuur gedoken
- Inventarisatie moment voortgang project
- Iinitatie gedaan tot afspraak voorlopig assesment
- RUP vs AUP
- Onderzoek is niet wetenschappelijk maar meer constatatering / literatuur onderzoek
- AUP fasering
- Inception → PVA
- Elaboration → Definitiestudie
- Construction → Code / Ontwerp
- Transition → Oplevering / Handleidingen
- Hierbij is overwogen
- De focus van AUP op definitieeren
- De focus van AUP op minder formeel ontwerpen
- Interview doelen opgesteld
- Sync items bepalen
- Procedures bepalen
- Risicos bepalen
- Werkmethodiek bepalen
- Groeps interview met engineers
- Arno is netwerk architect, voornamelijk verantwoordelijk voor het (hardware) ontwerp. Lucis is de software man, en deelt de functie met Arno
- Het netwerk is in continue transitie. Eerst van STP naar VLAN segregated, nu naar MPLS.
- Verdere achtergrond informatie is digitaal verstrekt

23-06
- Gantt kaart gemaakt van het project
- Kleur gecodeerd voor duidelijkheid.
- Project ingericht naar AUP
- Deels project, deels proces kaart, beide velden zijn erin opgenomen
- (foto Gantt kaart)

24-06
- Gewerkt aan procesverslag en opzetten logboek
- Formele start onderzoek.

27-06
- Samenvatten onderzoek tot dan toe
- Interview sales verwerkt
- Groeps interview verwerkt en aangevuld

- Achtergrond informatie verder uitgediept
- GBP kennis verbeterd
- Intranet doorzocht
- Database gedumpt en de opdrachtgever voor verduidelijking gevraagd
- Voortgang beheerd dvm aanvulling Gantt kaart
- Uitnodiging voorlopig assesment verstuurd
- Interview resultaten
- Procedures waarbij inconsistencies kunnen ontstaan
- Opruimen van poorten / patches
- Het toewijzen van poorten
- Bij klant wijzigingen
- Hernoemen
- Verhuizing
- Overname
- Vlans die vrij worden gegeven
- AS objecten die wijzigen
- Feedback over de voortgang komt niet van de opdrachtgever, heb vragen gesteld of de opdrachtgever hier wel voldoende tijd voor beschikbaar heeft.
- Zijn de consistency problemen technisch of procedureel?
- Een technische oplossing neemt slechts een deel van de problemen weg. Er is weinig tot geen draagvlak voor een procudurele oplossing van de problemen.

28-06
- Gesprek met opdrachtgever gehad
- 4e week komt er een draft elaboration
- Bevat een voorlopige indicatie van de velden die gesynchroniseerd moeten worden
- Impact van de velden bepalen.
- Waar zit het gevaar?
- Wat is de trigger waarde van een synchronisatie?
- Wat is het risico per veld?
- Frequentie bepalen
- Bepaling dat de volgende informatie beschikbaar moet zijn
- IST, SOLL, item, impact, frequentie (alles samen vormt risico)

29-06
- Interviews verder uitgewerkt
- Lijst met sync items opgesteld
- Database door Jan laten visualiseren voor verduidelijking
- Twee weken vakantie
- De manier van trend watching bepalen
- Bepalen hoe dit relateerd tot bijv Nagios of Cacti

12.11 MySQL Workbench installatie script

```
#Installing MySQLWorkbench 5.2.34 on Debian GNU/Linux 6.0

#Install required libraries
sudo apt-get install libsqlite3-dev libsqlite0-dev libmysqlclient-dev build-essential autoconf automake libtool libzip-dev libxml2-dev libsigc++-2.0-dev libglade2-dev libgtkmm-2.4-dev libgl1-mesa-dev uuid-dev liblua5.1-dev libpcre3-dev g++ libgnome2-dev libgtk2.0-dev libpango1.0-dev libcairo2-dev python-dev libboost-dev libctemplate-dev

#Get files, untar and browse to
cd ~

#Download in GUI: http://dev.mysql.com/get/Downloads/MySQLGUITools/mysql-workbench-gpl-5.2.34-src.tar.gz/from/http://mirror.leaseweb.com/mysql/
tar -xvzf mysql-workbench-gpl-5.2.34-src.tar.gz
cd mysql-workbench-gpl-5.2.34-src/

#Set path and compile (on 4 threads)
./autogen.sh

#Change path:
make -j4 install DESTDIR=~/mysql-workbench
make clean

#cleanup
cd ~ rm -Rf mysql-workbench-gpl-5.2.34-src*

#start
.~/mysql-workbench/
```

12.12 Schrijfplan procesverslag

Te volgen stappen:

- Wijzigingen van het document bijhouden met versienummer
- Logboek bijhouden gedurende het gehele traject, bevat:
    - Keuzemomenten
    - Proces verloop
    - Aannames
    - Bevindingen
    - Registratie werkzaamheden
- Logboek verwerking dmv:
    - Digitalisering notities en schetsen
    - Tabellen maken van beslissingsmomenten
    - Criteria opstellen
    - Keuzes toelichten
    - Kleur coderen logbestanden om doorzoekbaarheid te verbeteren
- Literatuur onderzoek naar vergelijkbare procesverslagen.
- Opstellen documentstructuur nav projectmethodiek
- Per fase invulling geven aan de beschrijving op de volgende manier
    - Ruwe body schrijven
    - De tekst volledig uitschrijven
    - De tekst tot een geheel samenbrengen en overbodige informatie eruit halen.
- De evaluaties uitwerken met de hierboven beschreven methode.
- De competenties bewijzen nav de bewijslast
- De literatuurlijst samenstellen
- Woordenlijst opstellen
- Bijlagen toevoegen
- Proofread sessie door collega
    - Controle op het verloop van de tekst
    - Controle op inhoudelijke fouten
- Proofread door leek
    - Controle op het verloop van de tekst
    - Controle op spellingfouten.