rijksuniversiteit groningen / faculteit wiskunde en natuurwetenschappen

Faculteit Wiskunde en Natuurwetenschappen

Master in Computing Science

Academiejaar 2011 – 2012

Een toekomstgericht testplatform voor Dienst Uitvoering Onderwijs

door

Johan van der Geest en Mark Ettema

6 juli 2012

Eerste begeleider RUG: prof. dr. Alexandru C. Telea

Tweede begeleider RUG: prof. dr. ir. Marco Aiello

Begeleiders DUO: Willem Hobers en Arjen Wassink

Dienst Uitvoering Onderwijs
Ministerie van Onderwijs, Cultuur en Wetenschap

Voorwoord

Bij DUO *(Dienst Uitvoering Onderwijs) hebben wij in de periode van februari tot juli 2012 ons afstudeeronderzoek uitgevoerd. Wij waren op zoek naar een uitdagende opdracht waarbij wij de kennis die wij tijdens onze opleiding hebben opgedaan goed konden toepassen. De grootte van de organisatie en complexiteit van de opdracht zorgde zeker voor de nodige uitdaging! Wij zijn erg trots op het behaalde resultaat en willen daar graag een aantal mensen voor bedanken.*

Wij willen onze begeleiders van DUO *en de* RUG *bedanken voor de tijd en moeite die zij in het afstudeeronderzoek hebben gestoken. Arjen Wassink en Willem Hobers hebben ons erg goed bij* DUO *geïntroduceerd en stonden altijd paraat voor feedback. Alex Telea was vanaf het eerste moment bij het onderzoek betrokken en heeft ons geïnspireerd met zijn kennis over het onderwerp. Wij zijn Alex Telea erg dankbaar voor de feedbackmomenten en de adviezen die hij ons heeft gegeven. Wij willen ook Marco Aiello bedanken voor zijn feedback als tweede begeleider.*

Tot slot willen wij de medewerkers van DUO *bedanken die aan ons onderzoek hebben meegewerkt, met in het bijzonder Robbert Jan van Meenen, Andries Mesken, Marko Ketellapper, Meeuwes Mollinger, Wout Hoekstra, Dorien Donker en Nico Bloem. Bedankt voor jullie input, feedback en begeleiding!*

Johan van der Geest en Mark Ettema, juli 2012

Samenvatting

Tijdens deze afstudeeropdracht is onderzoek gedaan naar een toekomstgericht testplatform voor DUO, waarin Java, een SOA (Service Oriented Architecture) en de Scrum ontwikkelmethodiek centraal staan. De huidige situatie, gewenste situatie, architectuur, tooling, beheersaspecten en evolutie van het testplatform zijn onderzocht. Ook is er een validatie bij de stakeholders uitgevoerd.

In de eerste fase van het onderzoek is de huidige situatie geanalyseerd. Uit gesprekken met medewerkers kwamen verschillende problemen en verbeterpunten naar voren. De meest belangrijke zijn: er is vrijwel geen overzicht van het testplatform, de overzettingen tussen omgevingen verlopen problematisch, de testomgevingen zijn te veel gebaseerd op de watervalmethodiek, het beheer wordt niet door één partij uitgevoerd, niemand voelt zich verantwoordelijk voor de testomgevingen, er wordt veel handmatig werk uitgevoerd, er is weinig koppeling tussen de verschillende systemen en veel communicatie verloopt via e-mail.

Vervolgens is de gewenste situatie onderzocht. Met behulp van requirements engineering zijn de stakeholders, user stories, key drivers, functionele en niet-functionele requirements in kaart gebracht. De vier key drivers – overzicht, automatisering, beschikbaarheid en gebruiksvriendelijkheid – hebben een grote invloed op de requirements, architectuur en ontwerpkeuzes van het testplatform.

De architectuur is met het "4+1" view model in kaart gebracht. Het testplatform bestaat uit third-party tooling en een deel maatwerksoftware, dat de werktitel TPM (Testplatform Manager) heeft. Via TPM krijgen gebruikers inzicht in het testplatform, zoals de aanwezige projecten, testomgevingen, softwarecomponenten (bestanden, klassen en functies) en relaties hiertussen. TPM automatiseert ook veel processen, zoals het aanvragen en overzetten van testomgevingen. De omgevingen draaien op een private cloud en zijn daardoor eenvoudig schaalbaar.

Het beheer moet onder een nieuw cluster vallen, met als werktitel TPB (Testplatform Beheer). Met behulp van SLA's, duidelijke functieomschrijvingen, verantwoordelijkheden en 1e-, 2e- en 3e-lijns support moet de kwaliteit van het beheer omhoog gaan. Eén testomgeving kan door meerdere Scrum projectrollen worden gebruikt, waardoor minder overzettingen nodig zijn.

Dit onderzoek heeft een brede scope. Vervolgonderzoek is nodig om meer de diepte in te gaan, alvorens de realisatie door DUO of een andere partij kan worden uitgevoerd. De validatiegesprekken waren erg positief. De feedbackpunten hiervan zijn in de scriptie opgenomen. Het onderzoek is een succes dat voor zowel DUO als de afstudeerders erg leerzaam is geweest.

Inhoudsopgave

1 Inleiding

Deze scriptie begint met een introductie van het bedrijf waar dit afstudeeronderzoek heeft plaatsgevonden. De missie, kernwaarden, afdelingen, infrastructuur en architectuur van DUO (Dienst Uitvoering Onderwijs) zijn behandeld. Vervolgens is de probleemstelling en de onderzoeksvraag beschreven. Het hoofdstuk eindigt met een overzicht van de inhoud van de scriptie.

1.1 Dienst Uitvoering Onderwijs

DUO is de uitvoeringsorganisatie van de Rijksoverheid voor het onderwijs. De missie van DUO is het financieren en informeren van onderwijsdeelnemers en onderwijsinstellingen en het organiseren van examens. Daarmee wil DUO helpen om het de professionals in het onderwijsveld makkelijker te maken, zodat deze meer tijd aan hun kernactiviteit, het onderwijs, kunnen besteden. Ze is de deskundige schakel tussen leerlingen, studenten, ouders, docenten, scholen, universiteiten, beleidsmakers, gemeenten en andere ketenpartners. De organisatie kent de volgende kernwaarden:

- **klantgericht**: zorgen voor duidelijkheid en gemak voor klanten en collega's
- **betrouwbaar**: een transparante organisatie zijn die doet wat ze zegt door concrete afspraken te maken waar de organisatie op afgerekend kan worden
- **toekomstgericht**: de dienstverlening blijven verbeteren en daarbij niet alleen naar de huidige wensen kijken, maar ook naar de toekomst
- **verbindend**: samenwerken en het hebben van een schakelfunctie tussen verschillende partijen.

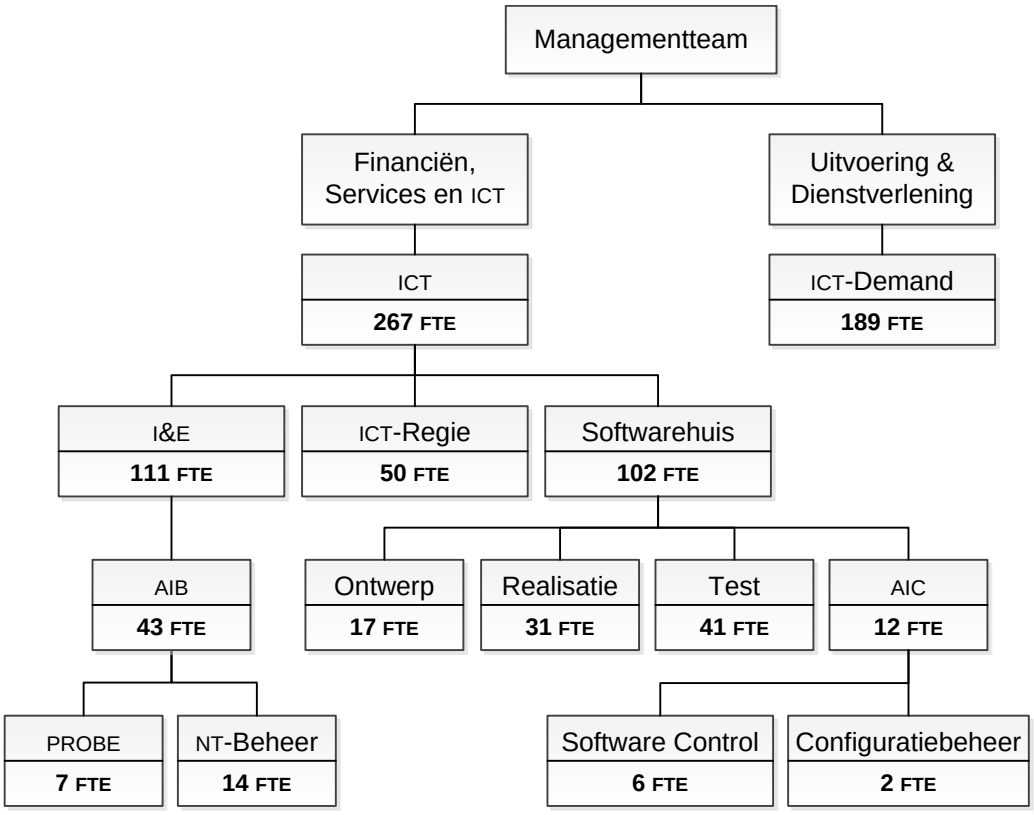
DUO is gevestigd in Groningen en Zoetermeer en heeft een aantal servicekantoren in het land. In Groningen bevindt zich het hoofdkantoor en een aantal andere vestigingen. De organisatie heeft een formatieomvang van circa 1800 FTE (Fulltime-Equivalent). Ongeveer een vierde van deze medewerkers heeft ICT gerelateerde taken. Per jaar worden er circa 300.000 ICT projecturen gemaakt.

DUO ontwikkelt veel software zelf. Het grootste softwaresysteem, WSF (Wet Studiefinanciering), stamt uit 1986 en wordt gehost in Apeldoorn. Het is geschreven in COBOL en telt ongeveer 1,5 miljoen LOC (Lines of Code). De toekomstvisie van DUO is om grote losstaande applicaties, vaak geschreven in oude programmeertalen, te vervangen door opgesplitste diensten in een SOA (Service Oriented Architecture). Er is voor Java gekozen als toekomstgerichte programmeertaal.

1.1.1 Afdelingen

De meeste organisaties zijn onderverdeeld in afdelingen, zo ook DUO. In figuur 1.1 staat een vereenvoudigd organogram. De functies van de afdelingen staan in tabel 1.1 beschreven. Het organogram bevat alleen de afdelingen die relevant zijn voor het onderzoek. Er zijn twee hoofdtakken te herkennen:

- **Financiën, Services en ICT**: de afdelingen onder deze tak leveren ondersteuning aan het primaire proces en richten zich op het verhogen van efficiency, het verlagen van de administratieve lasten en het leveren van kwaliteit
- **Uitvoering & Dienstverlening**: de afdelingen onder deze tak zijn verantwoordelijk voor de uitvoering van de hoofdtaken, waaronder het financieren van onderwijsinstellingen en -deelnemers en het organiseren van examens.



Figuur 1.1: Vereenvoudigd organogram van de organisatie. Alleen de voor het onderzoek relevante afdelingen zijn weergegeven.

Afdeling	Functie
ICT-Demand	Ondersteunt de bedrijfsprocessen door te voorzien in de informatiebehoefte en het geven van advies. De afdeling is verantwoordelijk voor de acceptatie van zowel nieuwe als gewijzigde programmatuur en het geven van ondersteuning aan de gebruikers.

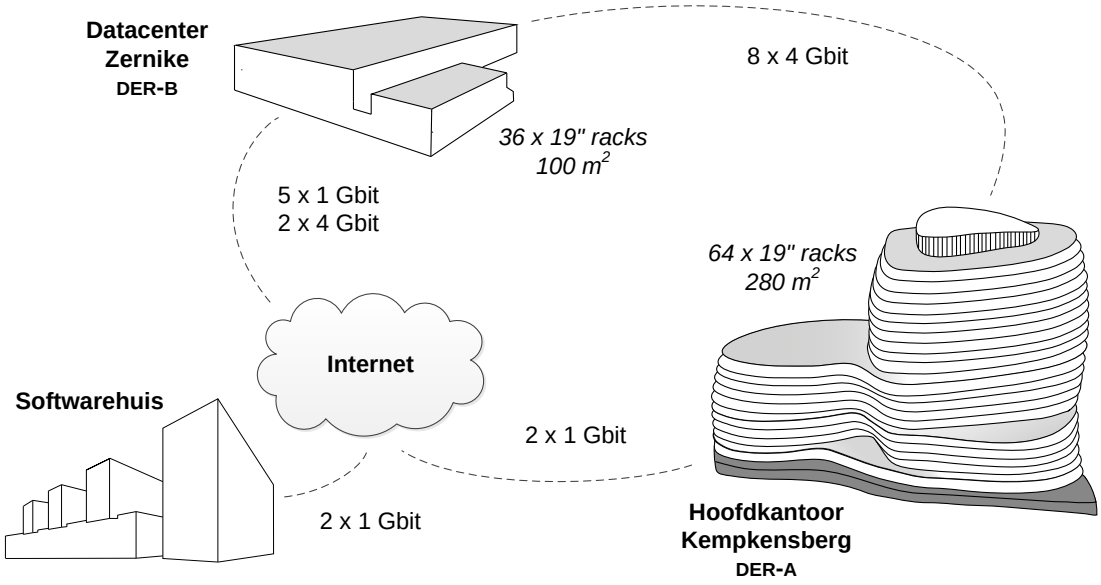
Afdeling	Functie
I&E	I&E (Infrastructuur & Exploitatie) is verantwoordelijk voor de ontwikkeling, instandhouding en het beheer van de technische infrastructuur en voor de implementatie en exploitatie van de systemen hierop.
· Vakgroep AIB	AIB (Applicatief Infrastructuur Beheer) houdt zich bezig met het technisch beheer in de laag boven het besturingssysteem.
· PROBE	PROBE (Productiebegeleiding) verzorgt de productiebegeleiding en exploitatie op het AS/400 platform (een midrangesysteem van IBM).
· NT-beheer	Verzorgt het applicatief technisch beheer op het Windows platform.
ICT-Regie	De leverancier en regisseur van projectleiding, SLM (Service Level Management) en contractmanagement.
Softwarehuis	Analyseert, ontwerpt, ontwikkelt en test maatwerksoftware.
· Vakgroep Ontwerp	Levert functioneel ontwerpers en beheert de door hen opgeleverde producten.
· Vakgroep Realisatie	Levert softwareontwikkelaars en geeft de business en ICT advies en opleiding in software-ontwikkelprocessen en tooling.
· Vakgroep Test	Levert testcoördinatoren en testers. Geeft technische ondersteuning voor het uitvoeren van tests en inrichten van testomgevingen en advies over testgerelateerde onderwerpen.
· Vakgroep AIC	De vakgroep AIC (Architectuur, Integratie en Configuratiebeheer) geeft de gewenste ontwikkelrichtingen aan, heeft technische kennis over de communicatie tussen platformen en vormt de kern van het SCC (SOA Competence Center). Het SCC geeft sturing en ondersteuning voor de SOA.
· Software Control	Verzorgt de administratie van alle softwarewijzigingen die worden opgeleverd door het Softwarehuis. Zorgt er verder voor dat de verschillende clusters bij I&E over de juiste informatie beschikken om gewijzigde software te installeren.
· Configuratiebeheer	Ondersteunt het testbevindingenbeheer en is verantwoordelijk voor het beheer van de configuratiedatabase en de applicatielandschapsvisualisaties.

Tabel 1.1: Een overzicht van de relevante afdelingen voor dit onderzoek en de bijhorende functieomschrijvingen.

1.1.2 Infrastructuur

Bij de ICT-dienstverlening van DUO wordt veel data getransporteerd; dit gebeurt zowel intern als naar de buitenwereld toe. Hiervoor beschikt de organisatie over twee datacenters in Groningen. Figuur 1.2 geeft een globaal overzicht van de infrastructuur. De relevante locaties voor dit onderzoek en de verbindingen tussen deze locaties staan hierin aangegeven. Andere locaties van DUO (Zoetermeer, Apeldoorn en de servicekantoren) vallen buiten de scope van het onderzoek en zijn daarom niet aangegeven.

Het primaire datacenter (DER-A) bevindt zich in het hoofdkantoor. Het tweede datacenter (DER-B) wordt gedeeld met andere bedrijven en bevindt zich op het Zernikecomplex. Dit datacenter wordt gebruikt als uitwijkcentrum. Indien er calamiteiten zijn in het primaire datacenter kan hier zonder dat de klanten en gebruikers het merken naar worden overgeschakeld. Beide datacenters zijn uitgerust met klimaatbeheersing, geavanceerde automatische brandblussystemen, back-up stroomvoorzieningen (UPS) en fysieke veiligheidsmaatregelen.

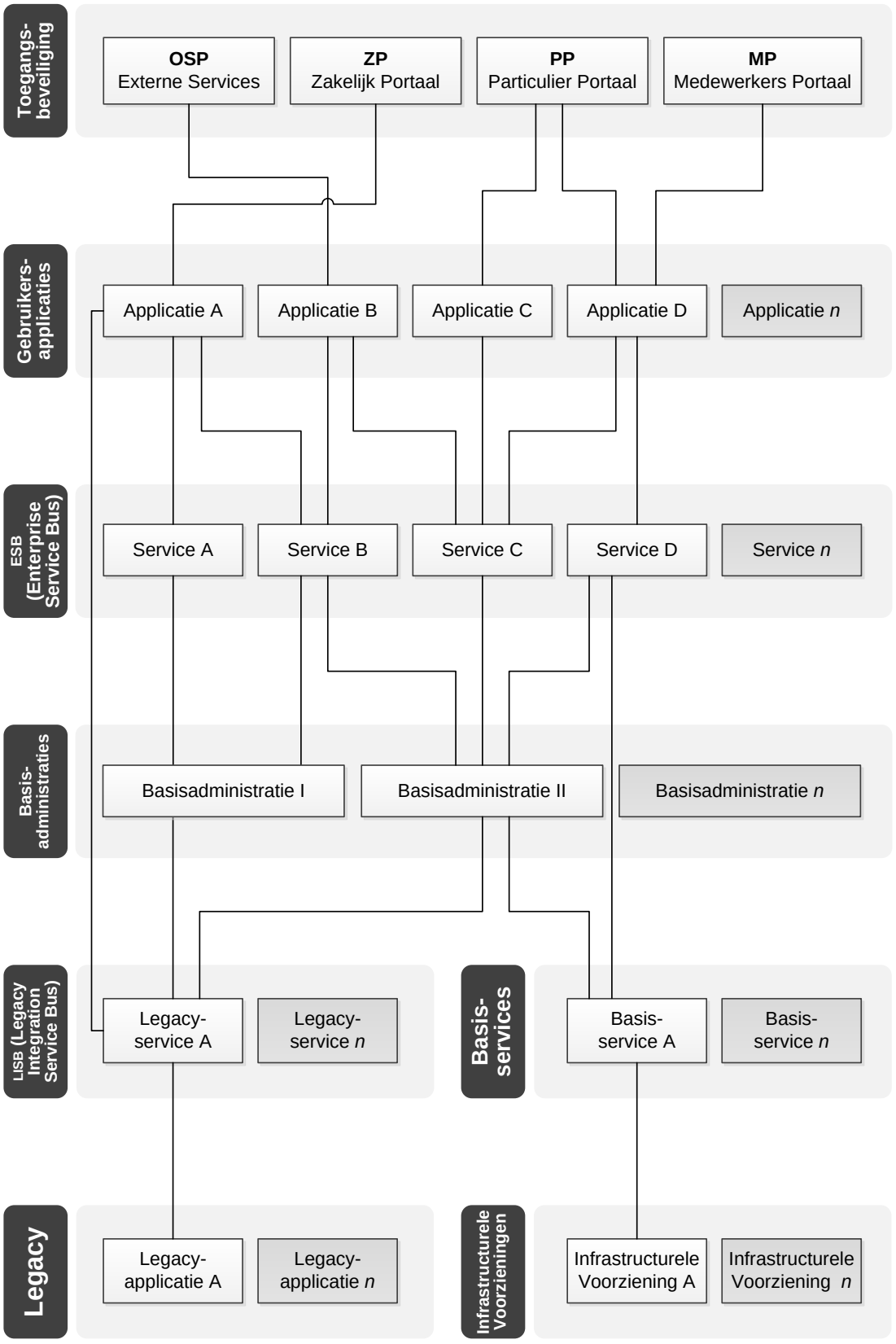


Figuur 1.2: Globaal overzicht van de infrastructuur in Groningen. De relevante locaties en de verbindingen hiertussen staan aangegeven.

1.1.3 Architectuur

DUO heeft een groot aantal softwaresystemen in beheer. Veel van deze systemen communiceren met elkaar, waardoor afhankelijkheden ontstaan. Een duidelijke architectuur is belangrijk om het geheel goed te kunnen beheren. In figuur 1.3 zijn de verschillende lagen van de softwarearchitectuur en de verbindingen hiertussen weergegeven.

DUO heeft in 2005 gekozen voor een SOA in combinatie met een ESB (Enterprise Service Bus). Hierbij worden gegevens en functionaliteiten beschikbaar gesteld via services met gestandaardiseerde interfaces. De ESB zorgt voor intelligente communicatie, waardoor services op een uniforme manier kunnen worden aangeroepen. De aanroeper hoeft geen kennis te hebben van de locatie, het platform en de technologie van de service [1]. DUO gebruikt Cordys als platform voor de ESB en IBM WebSphere als Java applicatieserver. De ESB is in de derde laag van figuur 1.3 weergegeven.



Figuur 1.3: De SOA in combinatie met een ESB, waarbij door de lagen een duidelijke scheiding tussen de verschillende componenten is te zien.

De toegangsbeveiligingslaag zorgt voor de beveiliging van de toegang tot de applicaties. Alle communicatie met de buitenwereld verloopt via de portalen of externe services. De authenticatie en autorisatie is op deze manier centraal geregeld. De gebruikersapplicatieslaag bevat de applicaties die door de eindgebruikers gebruikt worden. Deze applicaties communiceren met de basisadministraties via services die op de ESB worden aangeboden. De basisadministraties bevatten gegevens waar de gebruikersapplicaties gebruik van maken.

De laag onder de basisadministraties is opgesplitst in twee delen. De LISB (Legacy Integration Service Bus) zorgt voor de communicatie met de legacysystemen. Dit zijn oude systemen die geschreven zijn in COBOL, COOL:2E en COOL:PLEX. Een voorbeeld hiervan is WSF. Door de LISB zijn de systemen toch als service te benaderen. De andere helft van deze laag bevat de basisservices. Dit zijn services voor de communicatie met de infrastructurele voorzieningen, zoals het DWH (Datawarehouse).

In enkele gevallen wordt de architectuur doorbroken, omdat er legacysystemen zijn waarvoor nog geen service op de ESB beschikbaar is. Een gebruikersapplicatie communiceert in dat geval direct met een service in de LISB.

1.2 Probleemstelling

In de vorige paragrafen is beschreven dat DUO software ontwikkelt in Java, COBOL, COOL:Plex en COOL:2E. Alleen Java zal als programmeertaal moeten overblijven. Legacysystemen geschreven in andere talen zullen dan vervangen moeten zijn door nieuwe services en applicaties ontwikkeld in Java. Het aantal legacysystemen ligt nu rond de 40, met een totaal van circa acht miljoen LOC. Voor Java ligt het aantal rond de 60, met circa 660.000 LOC.

Het testen van software in COBOL, COOL:Plex en COOL:2E verloopt volgens DUO erg stabiel. De organisatie heeft meer dan 20 jaar ervaring met deze programmeertalen, de platformen waarop de software draait en de testomgevingen, tooling en processen die nodig zijn om de software gestructureerd te testen.

Definitie van Testomgeving

Een testomgeving is de combinatie van hardware (zoals een server of virtuele server) en software (zoals het besturingssysteem, frameworks, libraries en databases) waarop een test kan worden uitgevoerd.

In tegenstelling tot de legacytalen, vormt het testen van Java-software een knelpunt bij softwareprojecten bij DUO. De volgende problemen worden herkend:

- **beschikbaarheid**: geen tijdige beschikking over een werkende testomgeving
- **traceerbaarheid**: door de opsplitsing van grote applicaties in services en de afhankelijkheden die daardoor ontstaan is het lastiger om problemen te traceren
- **omgevingbeheer**: er is geen integraal beeld van wie wanneer gebruik wil maken van een testomgeving
- **versiebeheer**: er is geen overzicht van welke versies van de (gedeelde) componenten en (gedeelde) data in een omgeving aanwezig zijn
- **verantwoordelijkheid**: het is onduidelijk wie verantwoordelijk is voor het beheer van de testomgevingen.

Deze problemen leiden tot vertraging bij veel softwareprojecten. Dit heeft als gevolg dat er onnodige kosten worden gemaakt en het weerhoudt ontwikkelaars en testers ervan om met hun kerntaken bezig te zijn. Het PID (Project Initiatie Document) gaat verder in op de probleemstelling en de aanpak (zie bijlage A).

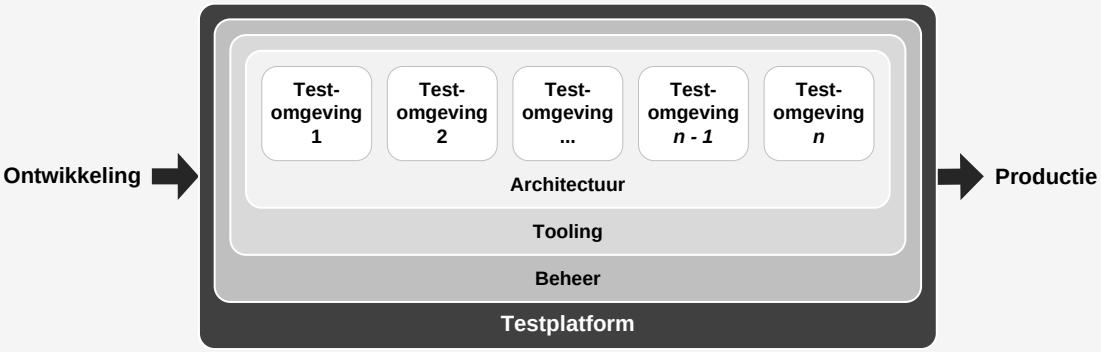
1.3 Onderzoeksvraag

Naar aanleiding van de probleemstelling is de onderzoeksvraag gedefinieerd. Deze geeft weer wat er tijdens het onderzoek onderzocht gaat worden. De onderzoeksvraag luidt:

> *"Hoe kan het testplatform van* DUO *het beste worden ingericht, zodat het voldoet aan de toekomstvisie waarin Java, een* SOA *en de Scrum ontwikkelmethodiek centraal staan?"*

Definitie van Testplatform

Een testplatform omvat de testomgevingen en de ondersteunende architectuur, tooling en beheersaspecten om het testproces te faciliteren. Het vormt de schakel tussen de ontwikkeling van software en het in productie nemen van deze software.



Gezien de omvang van de onderzoeksvraag zijn de volgende ondersteunende vragen opgesteld:

- Hoe wordt in de huidige situatie Java-software ontwikkeld en getest en welke problemen komt men hier bij tegen?
- Welke wensen (vertaald naar requirements) zijn er voor een toekomstgericht testplatform?
- Hoe moet de architectuur van het testplatform eruitzien om aan de requirements te voldoen?
- Welke tooling is er op de markt, reeds aanwezig of moet zelf worden ontwikkeld om de requirements en architectuur te ondersteunen?
- Welke beheersaspecten zijn nodig om de resterende niet-functionele requirements zoveel mogelijk te dekken?
- Met welke vervolgstappen kan het testplatform verder evolueren?
- In hoeverre komen de uitkomsten van het onderzoek overeen met de wensen van de belanghebbenden en worden deze geaccepteerd?

1.4 Opbouw van het document

De scriptie bestaat uit de volgende hoofdstukken:

- **Huidige situatie**: Hoofdstuk 2 beschrijft de huidige situatie. Er is onderzoek gedaan naar de huidige manier van ontwikkelen en testen van Java-software.
- **Gewenste situatie**: De stakeholders, user stories, key drivers en functionele en niet-functionele requirements voor een toekomstgericht testplatform staan in hoofdstuk 3 beschreven.
- **Architectuur**: Hoofdstuk 4 behandelt de architectuur van het testplatform, die met behulp van het "4+1" view model is beschreven.
- **Tooling**: Hoofdstuk 5 gaat in op de tooling die nodig is om de requirements en architectuur te ondersteunen.
- **Beheer**: De beheeraspecten van het testplatform, zoals het inrichten van de testomgevingen en de beherende partij, staan in hoofdstuk 6 beschreven.
- **Systeemevolutie**: Hoofdstuk 7 geeft antwoord op de vraag welke vervolgstappen nodig zijn na dit onderzoek.
- **Validatie**: Hoofdstuk 8 bevat de validatie van de onderzoeksresultaten die bij de belanghebbenden van dit onderzoek is uitgevoerd.
- **Conclusie**: Hoofdstuk 9 bevat de conclusie, die een antwoord geeft op de onderzoeksvraag.

De gebruikte afkortingen en begrippen zijn beschreven in de afkortingen- en begrippenlijst op pagina 97.

2 Huidige situatie

Dit hoofdstuk analyseert de huidige situatie van het ontwikkelen en testen van Java-software. De informatie voor deze analyse is verkregen via het intranet, beschikbare documentatie en gesprekken met verschillende DUO-medewerkers (zie bijlage B voor de gesprekkenlijst).

In de eerste paragraaf is het gebruikte platform beschreven. Daarna volgen twee paragrafen over de ontwikkelstraat (een term die door DUO wordt gebruikt voor het traject dat de software doorloopt van ontwerp tot productie). Tot slot volgt een paragraaf over de gebruikte tooling en een paragraaf met de problemen en verbeterpunten die de huidige situatie met zich meebrengt.

2.1 Platform

DUO gebruikt Java EE (Java Platform, Enterprise Edition) als ontwikkelplatform voor Java-software. Sinds 2009 wordt vrijwel alle nieuwe software ontwikkeld in versie 5 van Java EE. Van deze versie is dan ook uitgegaan bij het beschrijven van de huidige situatie. Daarvoor werd J2EE 1.4 gebruikt. Het Softwarehuis heeft voor elke gebruikte versie van Java EE richtlijnen vastgesteld die beschrijven welke frameworks en tooling gebruikt dienen te worden. In deze paragraaf zijn de applicatieservers, databases, message queue en de gebruikte ESB beschreven.

2.1.1 Applicatieservers

Een applicatieserver voorziet software van diensten zoals beveiliging, dataservices, transactie-ondersteuning, load balancing en het beheer van grote gedistribueerde systemen. Bij DUO wordt gebruikt gemaakt van IBM WAS (WebSphere Application Server) [2] versie 6.1. Deze voldoet aan de Java EE standaard. Dit is een Java-standaard waaraan een aantal voor bedrijfsdoeleinden belangrijke onderdelen is toegevoegd, zoals centrale gegevensbenadering en queue-koppelingen. Bij DUO draaien de applicatieservers op virtuele (VMWare) Windows servers, waarbij het technisch beheer wordt uitgevoerd door NT-Beheer.

Omdat er aanvankelijk geen developerversie van IBM WebSphere beschikbaar was en een volledige versie te hoge kosten met zich mee bracht, maakt DUO ook gebruik van de opensource Glassfish applicatieserver [3]. Ontwikkelaars gebruiken Glassfish om lokaal op hun werkstation te kunnen testen. Dit toont tevens aan dat de software applicatieserveronafhankelijk is. Glassfish voldoet aan de Java EE

standaard, waardoor de applicatieservers in theorie goed te combineren zijn. In de praktijk blijkt dit complexer. Inmiddels is van WebSphere een developerversie beschikbaar en wordt onderzocht of het efficiënter is om Glassfish te vervangen.

2.1.2 Databaseserver

In de huidige architectuurvoorschriften van DUO staat beschreven dat primaire data moet worden opgeslagen op de AS/400. In het geval van Java communiceren de applicaties met DB2-databases op de AS/400. Dit is een relationele database ontwikkeld door IBM. Het product is beschikbaar voor een groot aantal besturingssystemen. DUO heeft veel ervaring met het AS/400 platform, waarop naast de databases nog een groot aantal legacyapplicaties draait. In Java wordt met behulp van JDBC (Java Database Connectivity) verbinding gemaakt met de database. PROBE is verantwoordelijk voor het technisch beheer van de AS/400 en daardoor ook voor de DB2-databases.

2.1.3 Message Queue

MQ (Message Queue) is een middleware transactiesysteem dat gegevens tussen systemen uitwisselt over hiervoor gecreëerde communicatiekanalen. Processen kunnen asynchroon met elkaar communiceren door een bericht in een queue te plaatsen. De zender en de ontvanger hoeven niet tegelijkertijd verbonden te zijn met de queue, want de MQ-queuemanager zorgt ervoor dat het bericht beschikbaar blijft tot dat deze is afgeleverd [4]. Uitwisseling is mogelijk met alle platformen waarop MQ geïnstalleerd kan worden. Een groot aantal platformen wordt ondersteund, waaronder Windows, de AS/400 en de IBM 3090. Bij DUO wordt gebruik gemaakt van IBM WebSphere MQ versie 6. De MQ-server draait op Windows en NT-Beheer is verantwoordelijk voor het technisch beheer.

2.1.4 Enterprise Service Bus

Bij het invoeren van de SOA is ervoor gekozen om gebruik te gaan maken van een ESB. Via een openbare aanbesteding is de ESB van Cordys geselecteerd. Deze software draait op Windows servers die worden beheerd door NT-Beheer. Daarnaast houdt de cluster Integratie van de vakgroep AIC zich intensief bezig met de ESB. Deze cluster beschikt over de kennis die nodig is om applicaties aan te sluiten op de bus. De ESB wordt gezien als infrastructureel component, waarop geen businesslogica wordt geïmplementeerd.

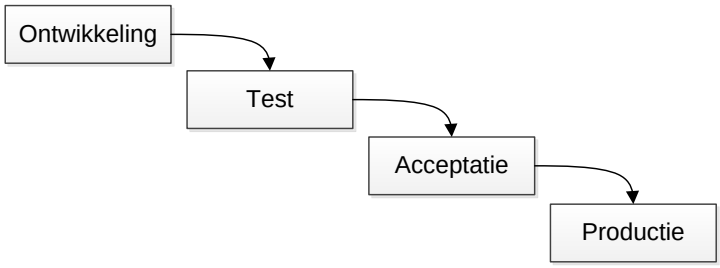
De ESB zorgt ervoor dat services via een centrale bus kunnen worden aangeroepen. Hierbij is het niet nodig om de locatie van een service te weten en ook protocolswitches worden automatisch uitgevoerd (bijvoorbeeld van HTTP naar MQ en andersom) [5]. De ESB kan kleine transformaties uitvoeren, zoals het omzetten van een correspondentienummer naar een burgerservicenummer. Daarnaast is monitoring, authenticatie en het beheer van de services centraal geregeld.

2.2 Ontwikkelstraat

Voorafgaand aan het testen wordt software ontwikkeld. Het is van belang deze stap te analyseren, omdat de testomgevingen deels gebaseerd zijn op de keuzes die voor de ontwikkeling van software zijn gemaakt. Bij DUO wordt software ontwikkeld volgens de OTAP-methodiek. In figuur 2.1 staan de fasen hiervan weergegeven. De testaanpak is gebaseerd op TMAP (Test Management Approach) Next van Sogeti [6]. Het pad van ontwikkeling, test, acceptatie en productie heeft een sterke overeenkomst met die van de watervalmethodiek. De omgevingen binnen de ontwikkelstraat van DUO zijn gebaseerd op deze methodiek.

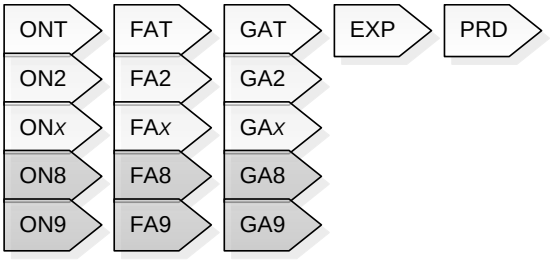
Definitie van Omgeving

Een omgeving is een samenstelsel van componenten zoals hardware, software, communicatiemiddelen, procedures en de faciliteiten voor het uitvoeren van een bepaald soort taken.



Figuur 2.1: DUO maakt gebruik van de OTAP-methodiek, waarbij een pad wordt doorlopen van ontwikkeling, testen, acceptatie naar productie.

De ontwikkelstraat bestaat uit een ONT (ontwikkel)-, FAT (functionele test)-, GAT (gebruikersacceptatietest)-, EXP (exploitatie)- en PRD (productie)- omgeving. Dit is één omgeving meer dan figuur 2.1 aangeeft, want acceptatie is bij DUO opgesplitst in een GAT- en EXP-omgeving. In de paragrafen 2.2.1 tot en met 2.2.5 zijn deze omgevingen beschreven. Om de productieomgeving te bereiken gaat de software de gehele straat door waarbij geen omgevingen kunnen worden overgeslagen. In figuur 2.2 staat de opbouw van de ontwikkelstraat afgebeeld.



Figuur 2.2: De huidige opbouw van de ontwikkelstraat. In de ONT-, FAT- en GAT-omgeving zijn parallelle straten mogelijk.

Parallelle straten maken het mogelijk om gelijktijdig verschillende versies van een applicatie te ontwikkelen en testen. Om risico's met betrekking tot de productieomgeving te vermijden zijn parallelle straten alleen toegestaan in de ONT-, FAT- en GAT-omgeving. In de EXP-omgeving moet het geheel zijn samengevoegd tot één versie. Van deze straten hebben er twee een speciale functie. De 8-straat bevat altijd een productieversie, dit om te zorgen dat van elke applicatie een productieversie beschikbaar is. De 9-straat wordt gebruikt voor het snel doorvoeren van een bugfix en wordt daarom ook wel de bugfixstraat genoemd.

Ondanks dat de ontwikkelstraat op de watervalmethodiek is gebaseerd wordt bij DUO ook ontwikkeld volgens de ontwikkelmethodieken RUP (Rational Unified Process) en Scrum. Hierbij worden dezelfde omgevingen gebruikt maar ziet de uitvoering van het ontwikkel- en testproces er anders uit.

Definitie van RUP

RUP staat voor Rational Unified Process en is een ontwikkelmethodiek voor een iteratieve en incrementele ontwikkeling van software. Het is gebaseerd op het analyseren van risico's en het verkleinen van deze risico's. Een RUP-proces bestaat uit het uitvoeren van cycli, waarbij elke cyclus eindigt met de oplevering van een nieuwe versie van het product.

Definitie van Scrum

Scrum is een Agile ontwikkelmethodiek dat zich primair richt op de business value [7]. Er wordt gewerkt in multidisciplinaire teams die in korte sprints (iteraties van twee tot vier weken) werkende software opleveren. Na iedere sprint kunnen de wensen en prioriteiten vanuit de business voor verdere ontwikkeling worden bijgesteld. Zie ook bijlage C.

In de komende paragrafen is per omgeving uitgelegd waar de omgeving voor dient, wie er verantwoordelijk voor zijn en welke taken binnen de omgeving worden uitgevoerd. De huidige manier van werken is beschreven met behulp van use cases. Als de manier van communiceren niet vermeld is, kan er vanuit worden gegaan dat e-mail wordt gebruikt. De gebruikte tools zijn toegelicht in paragraaf 2.4.

2.2.1 Ontwikkeling

Het bouwen van software wordt lokaal op de PC van de ontwikkelaar gedaan; dit wordt ook wel de pre-ONT genoemd. De ONT-omgeving is de eerste omgeving in de straat en wordt door de ontwikkelaars gebruikt om software te testen. De technische invulling van de ONT-omgeving kan afwijken van de productieomgeving. Daarmee wordt bedoeld dat de beschikbare resources kunnen afwijken en dat er andere applicaties op dezelfde server aanwezig kunnen zijn. In de ONT-omgevingen worden dan ook geen performancetests uitgevoerd. Het beheer wordt

gedaan door de softwareontwikkelaars zelf. Onderlinge afspraken moeten ervoor zorgen dat de ontwikkelaars elkaar niet tegenwerken bij het aanbrengen van wijzigingen.

Aanvraag omgevingen

Bij een nieuwe applicatie zullen de benodigde (test)omgevingen moeten worden aangevraagd. De aanvraag wordt gedaan bij Configuratiebeheer. Deze cluster overlegt over de juiste naamgeving, legt de applicatie vast in ITSM en geeft een applicatiecode terug. Deze applicatiecode wordt ook aan PROBE doorgegeven zodat, indien nodig, een DB2-database wordt aangemaakt. Via een e-mail kan een ONT-omgeving worden aangevraagd. Deze is meestal binnen een dag beschikbaar. De overige omgevingen moeten worden aangevraagd via een RFC (Request For Change). I&E zorgt voor de realisatie van deze omgevingen.

Bouwtest

Door de ontwikkelaar wordt een bouwtest uitgevoerd. Deze test heeft als doel na te gaan of de gerealiseerde software testbaar is en of het een correcte, complete en consistente uitwerking van de technische documentatie en technische eisen is. Afhankelijk van de gekozen ontwikkelmethodiek zal de hoeveelheid beschikbare documentatie verschillen. De bouwtest bestaat uit twee delen:

- **Programmatest:** Heeft als doel het aantonen van een juiste werking van de code. Hierbij worden de kleinst mogelijke testbare bouwstenen afzonderlijk getest, ook wel unittest genoemd.
- **Integratietest:** Heeft als doel het aantonen dat interfaces (via een API-call, MQ, database, tussenbestand of rechtstreeks) tussen de processen technisch aansluiten. Hiermee wordt nagegaan of het proces als geheel (de reeks afzonderlijk geteste programma's en processen samen) technisch gezien werkt.

Doordat de bouwtest onderdeel is van de ontwikkelfase valt deze buiten de scope van dit onderzoek. Zie ook de definitie van testplatform in paragraaf 1.3. Bij een vervolgonderzoek kan de scope worden verbreed (zie paragraaf 7.1).

Overzetten

Voordat van de ONT-omgeving gebruik wordt gemaakt, test de ontwikkelaar de software lokaal op zijn werkstation. Als dat conform de technische eisen werkt zal de ontwikkelaar de software overzetten naar de ONT-omgeving en ook hier gaan testen. De stappen die worden uitgevoerd voor de overzetting zijn beschreven in use case 2.1.

	Overzetten lokaal → ONT
Omschrijving	Het overzetten van de software van het werkstation van de ontwikkelaar (Glassfish) naar de ONT-omgeving (WAS).
Actor	Ontwikkelaar.

	Overzetten lokaal → ONT
Preconditie	De software is lokaal getest op de Glassfish applicatieserver.
Basisverloop	1. De ontwikkelaar onderzoekt of er aanpassingen op de ONT WAS server nodig zijn. Dit gaat eventueel in overleg met Software Control. 2. De ontwikkelaar zet de software over en controleert of deze technisch gezien werkt. 3. De ontwikkelaar geeft aan dat de software conform technische eisen werkt op de ONT-omgeving.
Alternatief verloop 1	*Er zijn wijzigingen nodig in de* ONT-*omgeving:* 2a. De ontwikkelaar voert de wijzigingen door en documenteert deze. 3a. *Ga verder met stap 2 van het basisverloop.*
Postconditie	De software werkt technisch correct op de ONT WAS.

Use case 2.1: Stappen die worden uitgevoerd voor de overzetting van het lokale werkstation naar de ONT-omgeving.

Met behulp van de "Checklist Bouwtest" controleert de ontwikkelaar of alle aspecten getest zijn. Zodra dit gereed is en de software stabiel in de ONT-omgeving draait kan de software door naar de functioneel tester. Dat gebeurt in de FAT-omgeving, dus de software moet worden overgezet. Hierbij wordt soms gebruik gemaakt van Jython-scripts. Daarmee kan een deel van de handmatige acties worden geautomatiseerd. De overzetting staat beschreven in use case 2.2.

Definitie van Checklist Bouwtest

De Checklist Bouwtest bevat de aspecten die door de ontwikkelaar getest moeten zijn en beschrijft enkele eisen waar deze tests aan moeten voldoen.

Definitie van Vrachtbrief

Een vrachtbrief, ook wel overzetdocument genoemd, wordt gebruikt voor overzettingen tussen twee opeenvolgende omgevingen in de ontwikkelstraat. Het document bevat naast de standaard gegevens (zoals de naam en applicatiecode) de volgende informatie:

- functionaliteit en versie die wordt opgeleverd
- de bevindingen die zijn opgelost
- bekende problemen
- of extra foutlogging (gedrag dat niet functioneel beschreven staat) is toegevoegd
- het buildnummer
- het stappenplan om de software draaiende te krijgen
- een overzicht van de voorgaande overzettingen.

	Overzetten ONT → FAT
Omschrijving	Het overzetten van de software van de ONT- naar de FAT-omgeving.
Actor	Ontwikkelaar.
Precondities	• De juiste versie van de software staat in de ONT-omgeving. • Het project is in Turnover aangemaakt.
Basisverloop	1. De ontwikkelaar vraagt een overzettaak aan bij PROBE voor het overzetten van de database. 2. De ontwikkelaar vult het formulier voor de Turnover-overzetting in. 3. De ontwikkelaar maakt een ZIP-bestand met de configuratiebestanden. 4. De ontwikkelaar beschrijft in de vrachtbrief de handmatige acties die uitgevoerd moeten worden om de software werkend te krijgen. 5. De ontwikkelaar stuurt de vrachtbrief (inclusief Turnoverformulier) naar de FAT-tester en naar Software Control. 6. Software Control keurt de aanvraag, in overleg met de testcoördinator en de testers, goed en stelt de uitvoerende partijen op de hoogte. 7. PROBE voert de automatische databaseoverzetting uit via Turnover. Daarna worden de eventuele handmatige acties op basis van de vrachtbrief uitgevoerd. Als de overzetting voltooid is wordt dit aan Software Control gemeld. 8. De ontwikkelaar voert de overzetting van de software uit en stelt Software Control hiervan op de hoogte. 9. Software Control handelt de administratieve taken af en informeert de betrokken partijen.
Alternatief verloop 1	*Er zijn geen wijzigingen in de configuratiebestanden aangebracht:* 3a. De ontwikkelaar voegt de overzetgegevens aan de vrachtbrief toe. 4a. *Ga verder met stap 5 van het basisverloop.*
Alternatief verloop 2	*Er zijn wijzigingen in de configuratiebestanden aangebracht:* 3b. De gewijzigde bestanden worden vergeleken met behulp van WinMerge. Hiervan worden DIFF-bestanden gemaakt waarvan samen met de configuratiebestanden een ZIP-bestand wordt gemaakt. 4b. De ontwikkelaar voert de wijzigingen in de vrachtbrief door. 5b. *Ga verder met stap 5 van het basisverloop.*
Alternatief verloop 3	*Er wordt van Jython-scripts gebruik gemaakt:* 4c. De ontwikkelaar zet de bestanden die nodig zijn voor de Jython-scripts klaar. 5c. De ontwikkelaar beschrijft de handmatige acties in de vrachtbrief en verwijst daarbij naar de bestanden die nodig zijn voor het uitvoeren van de Jython-scripts. 6c. *Ga verder met stap 5 van het basisverloop.*
Alternatief verloop 4	6d. Software Control keurt na overleg de aanvraag af en meldt dit terug aan de betrokken partijen.
Postconditie	De software is succesvol overgezet naar de FAT-omgeving en de bijbehorende administratieve taken zijn afgehandeld.

	Overzetten ONT → FAT
Alternatieve postconditie	De overzetting is afgekeurd en niet uitgevoerd.

Use case 2.2: Stappen die worden uitgevoerd voor de overzetting van de ONT- naar de FAT-omgeving.

2.2.2 Functionele test

In de FAT-omgeving wordt functioneel getest door de testers van het Softwarehuis. De technische invulling van deze omgeving kan afwijken van de productieomgeving. De testnavigatoren van de vakgroep Test van het Softwarehuis zijn verantwoordelijk voor het technisch beheer en geven ondersteuning. Er zijn afgeleide omgevingen (zie paragraaf 2.3.1), zodat er gelijktijdig kan worden getest.

Voor het testen kunnen er afhankelijkheden zijn met andere systemen. Deze systemen draaien in andere omgevingen, waardoor een koppeling moet worden gemaakt. Ook deze systemen kunnen afhankelijkheden hebben, waardoor een zogeheten keten ontstaat. De testnavigatoren kunnen deze koppelingen maken en houden in een Access-database een overzicht van deze koppelingen bij om te voorkomen dat projecten door koppelingen elkaar beïnvloeden.

Definitie van Keten

Een keten is een verzameling van actoren die zijn geordend volgens een logistiek proces. Deze aaneenschakeling van processen is gericht op het gezamenlijk bereiken van een resultaat. Bijvoorbeeld een service die een andere service nodig heeft om een juist resultaat terug te geven.

Systeemtest

Door de functioneel testers wordt een systeemtest uitgevoerd met als doel na te gaan of de gerealiseerde software functioneel correct werkt. Hierbij wordt zowel syntactisch (“accepteert een numeriek invoerveld alleen nummers?”) als semantisch (“wordt bij het selecteren van Nederland het postcodeveld actief?”) getest. De test bestaat uit drie delen:

- **FT (Functionele Test):** Heeft als doel na te gaan of de software een juiste uitwerking is van de functionele documentatie. Dit is een blackboxtest waarbij wordt gecontroleerd of de functioneel beschreven invoer leidt tot de functioneel beschreven uitvoer. De testcases worden door de tester met behulp van diverse tools (waaronder TFAT) en de beschikbare documentatie opgesteld. Bij het testen van een applicatie mogen alleen de gebruikers- en beheerinterfaces gebruikt worden. Bij een service mag er direct via de XML-interface getest worden.

- **GEIT (Geïntegreerde Test):** Heeft als doel na te gaan of de software het bedrijfsproces op de juiste manier ondersteunt. Het bedrijfsproces wordt van begin tot eind doorlopen, zodat ook de integratie met de reeds aanwezige software wordt getest. De test richt zich op de normale gevallen die veel voorkomen. Uitzonderingen zijn al voldoende met de FT getest.
- **Regressietest:** Het doel van deze test is controleren of alle ongewijzigde onderdelen van het systeem nog correct functioneren na het doorvoeren van een wijziging. Er wordt gebruik gemaakt van eerder opgestelde testcases uit het testdossier.

Definitie van Testdossier

Het testdossier is een door DUO ontwikkeld Excel document dat gebruikt wordt voor het vastleggen van testgevallen en de resultaten hiervan.

Bij de uitvoering van deze tests wordt gebruikt gemaakt van het testdossier in Excel. Met behulp van de invoegtoepassing LSD (Lokaal Snel Draaien) is het mogelijk een deel van deze tests automatisch uit te voeren. De tester geeft het verschil tussen het functioneel ontwerp en de programmatuur aan als bevinding. Bevindingen worden geregistreerd met de tool ClearQuest. Daarnaast kan de tester andere tools gebruiken (zie paragraaf 2.4) om het testen te vereenvoudigen. De stappen die een functioneel tester over het algemeen doorloopt staan beschreven in use case 2.3.

	Functioneel testen
Omschrijving	Het functioneel testen van de software.
Actor	Functioneel tester.
Preconditie	De juiste versie van de software staat in de FAT-omgeving.
Basisverloop	1. De functioneel tester bepaalt in overleg met de testcoördinator wat er getest moet worden en maakt hierbij gebruik van de beschikbare documentatie. 2. De functioneel tester voert (met behulp van tools) de verschillende testgevallen uit en geeft de resultaten aan in het testdossier. 3. De functioneel tester meldt aan de testcoördinator dat de test is afgerond en of er bevindingen zijn gedaan.
Alternatief verloop 1	3a. De functioneel tester doet een grote bevinding en registreert deze in ClearQuest. 4a. *Ga verder met stap 2 van het basisverloop.*
Alternatief verloop 2	3b. De functioneel tester doet een kleine bevinding en bespreekt dit met de ontwikkelaar of de ontwerper. 4b. *Ga verder met stap 2 van het basisverloop.*
Postconditie	De software is functioneel getest en (deels) functioneel gereed.

	Functioneel testen
Alternatieve postconditie	De software is functioneel getest, maar er zijn belangrijke bevindingen gedaan waardoor de software niet (deels) functioneel gereed kan worden gemeld.

Use case 2.3: Stappen die worden uitgevoerd door de functioneel testers.

Overzetten

Als de software (deels) functioneel gereed is, kan deze worden overgezet naar de GAT-omgeving voor de gebruikersacceptatietest. Functioneel gereed betekent dat alle bevindingen zijn afgesloten, waarbij het niet noodzakelijk is dat ook alle bevindingen zijn opgelost. Er kan voor worden gekozen om een bevinding in een volgende release op te lossen. De stappen die voor het overzetten van FAT naar GAT nodig zijn staan beschreven in use case 2.4.

	Overzetten FAT → GAT
Omschrijving	Het overzetten van de software van de FAT- naar de GAT-omgeving.
Actor	Testcoördinator.
Precondities	• De juiste versie van de software staat in de FAT-omgeving. • De software is (deels) functioneel gereed.
Basisverloop	1. De testcoördinator geeft aan welk deel van de software functioneel gereed is en meldt dit aan de GAT-tester en aan Software Control. *Alle programmatuur wordt altijd overgezet, maar slechts het gedeelte dat functioneel gereed is mag getest worden.* 2. De GAT-tester geeft bij Software Control aan dat de overzetting akkoord is en kan worden uitgevoerd. 3. Software Control voegt de overzetgegevens aan de vrachtbrief toe en stuurt deze (inclusief een Turnoverformulier) naar de uitvoerende partijen. 4. PROBE voert de automatische databaseoverzetting uit via Turnover. Daarna worden de eventuele handmatige acties op basis van de vrachtbrief uitgevoerd. Als de overzetting voltooid is, wordt dit aan Software Control gemeld. 5. De NT-Beheerder voert de Java-overzetting uit door het volgen van het stappenplan uit de vrachtbrief. In de praktijk wordt dit vaak samen met de ontwikkelaar gedaan, omdat deze meer Java kennis heeft. Als de overzetting voltooid is wordt dit aan Software Control gemeld. 6. Software Control zorgt ervoor dat de betrokken partijen worden geïnformeerd over de overzetting.
Alternatief verloop 1	2a. De GAT-tester geeft bij Software Control aan dat de overzetting nog niet mag worden uitgevoerd. 3a. Software Control laat de betrokken partijen weten dat de overzetting nog niet wordt uitgevoerd.

	Overzetten FAT → GAT
Postconditie	De software is succesvol overgezet naar de GAT-omgeving en de bijbehorende administratieve taken zijn afgehandeld.
Alternatieve postconditie	De overzetting is niet uitgevoerd.

Use case 2.4: Stappen die worden uitgevoerd voor de overzetting van de FAT- naar de GAT-omgeving.

2.2.3 Gebruikersacceptatietest

In de GAT-omgeving wordt getest op bruikbaarheid binnen het bedrijfsproces. Dit wordt gedaan door de (eind)gebruikers en functioneel beheerders. De technische invulling van de GAT-omgeving kan afwijken van de productieomgeving. De GAT-testcoördinator is verantwoordelijk voor de planning en bepaalt wie van welke omgeving gebruik maakt. Er zijn afgeleide omgevingen voor het gelijktijdig uitvoeren van tests.

Acceptatietest

De gebruikersacceptatietest heeft als doel na te gaan of de opgeleverde ondersteuning van de bedrijfs- en beheerprocessen voldoen aan de functionele en kwalitatieve eisen. In het geval van RUP of Scrum wordt de acceptatietest iedere iteratie uitgevoerd. In sommige gevallen zijn voor de test verbindingen met externe partijen (zoals onderwijsinstellingen) nodig. In dat geval wordt gebruik gemaakt van een VT (veldtest)-omgeving, die beschreven is in paragraaf 2.3.2.

ICT-Demand hanteert een stappenplan voor test- en acceptatieproces. Aan de hand van checklisten worden de acceptatiecriteria gecontroleerd op naleving en volledigheid. Een aantal tests wordt geautomatiseerd uitgevoerd. Dit gebeurt deels met eigengemaakte invoegtoepassingen voor Excel. Afhankelijk van datgene wat getest moet worden kan het zijn dat de verschillende testrollen (GAT-tester en GAT-testcoördinator) door dezelfde persoon worden uitgevoerd.

Bij de gebruikersacceptatietest wordt getest met een grotere dataset dan tijdens de functionele test. Deze dataset bevat een kopie van echte productiegegevens. De GAT-tester zou met behulp van het testdossier moeten weten wat er tijdens de functionele test is uitgevoerd. In de praktijk blijkt dit echter niet altijd het geval. In sommige gevallen wordt een belangrijk onderdeel van de functionele test nog eens herhaald met de grotere dataset. Over het algemeen gaat deze test sneller dan de functionele test. Het gaat er om of de gebruiker begrijpt hoe het werkt en of de functioneel beheerder de juiste beheergegevens eruit kan halen.

Na uitvoering van de test wordt een eindrapport opgesteld. Het uiteindelijke acceptatiebesluit wordt door de opdrachtgever van het project genomen. Een aantal

testproducten die zijn opgesteld gedurende de GAT kunnen worden hergebruikt bij een volgende test. Dit kunnen testontwerpen, geautomatiseerde testscripts en uitgangssituaties zijn. Use case 2.5 bevat de stappen die bij de gebruikersacceptatietest worden doorlopen.

	Gebruikersacceptatietest
Omschrijving	Het testen van de software op functionele en kwalitatieve eisen.
Actoren	GAT-tester.
Preconditie	De juiste versie van de software staat in de GAT-omgeving.
Basisverloop	1. De GAT-tester controleert of de opgeleverde producten (zowel de software als de documentatie) bij de start van de test van voldoende kwaliteit zijn zodat er zinvol getest kan worden. 2. De GAT-tester voert de test op basis van het testscript uit en beschrijft de uitkomsten. De test kan zowel uit handmatige als geautomatiseerde acties bestaan. 3. De GAT-tester beoordeelt het resultaat van de test en rapporteert dit aan de GAT-testcoördinator.
Alternatief verloop 1	*Er is nog geen testgevallenbeschrijving* GAT *beschikbaar:* 2a. De GAT-tester maakt de testgevallenbeschrijving GAT aan. 3a. *Ga verder met stap 2 van het basisverloop.*
Alternatief verloop 2	*Voor het uitvoeren van de test is een uitgangssituatie nodig:* 2b. De GAT-tester zet de juiste uitgangssituatie voor de test klaar. Dit gebeurt handmatig of met behulp van tools. 3b. *Ga verder met stap 2 van het basisverloop.*
Alternatief verloop 3	*Er is een bevinding gedaan:* 3c. De GAT-tester maakt in samenwerking met de GAT-testcoördinator een bevinding en registreert deze in ClearQuest. 4c. *Ga verder met stap 2 van het basisverloop.*
Postconditie	De software heeft de gebruikersacceptatietest succesvol doorlopen.
Alternatieve postconditie	De software heeft de gebruikersacceptatietest doorlopen, maar er zijn bevindingen gedaan waardoor de software nog niet gereed is bevonden.

Use case 2.5: Stappen die worden uitgevoerd voor de gebruikersacceptatietest.

Definitie van Testgevallenbeschrijving GAT

De testgevallenbeschrijving GAT is een sjabloon in Excel met een beschrijving bestaande uit algemene informatie (proces, onderdeel, versie, etc.), de testgevallenbeschrijving in logische zin, de fysieke testvulling (databaseinhoud) en het testscript (draaiboek, handelingen, etc.). In het sjabloon is specifieke gebruiksinformatie als toelichting opgenomen.

Overzetten

Als het acceptatiebesluit is genomen kan de software worden overgezet naar de exploitatieomgeving. Hierbij moet de software altijd als geheel worden overgezet. Dit in tegenstelling tot de voorgaande overzettingen, waarbij dat wel is toegestaan en alvast delen getest konden worden. Deze keuze is gemaakt omdat de uiteindelijke overzetting naar productie ook als geheel moet worden uitgevoerd. Op deze manier kan dit overzetproces alvast worden getest. Net als bij de overzetting vanuit de FAT-omgeving kan het zijn dat is besloten om een aantal bevindingen pas bij een volgende release op te lossen. De stappen die voor de overzetting nodig zijn staan beschreven in use case 2.6.

	Overzetten GAT → EXP
Omschrijving	Het overzetten van de software van de GAT- naar de EXP-omgeving.
Actor	GAT-testcoördinator.
Precondities	• De juiste versie van de software staat in de GAT-omgeving. • De software is geaccepteerd door de opdrachtgever van het project.
Basisverloop	1. De GAT-testcoördinator meldt aan Software Control dat de software kan worden overgezet naar de EXP-omgeving. 2. Software Control voegt de overzetgegevens aan de vrachtbrief toe en stuurt deze (inclusief een Turnoverformulier) naar de uitvoerende partijen. 3. PROBE voert de automatische databaseoverzetting uit via Turnover. Daarna worden eventuele handmatige acties op basis van de vrachtbrief uitgevoerd. Als de overzetting voltooid is, wordt dit aan Software Control gemeld. 4. De NT-Beheerder voert de Java-overzetting uit door het volgen van het stappenplan uit de vrachtbrief. In de praktijk wordt dit vaak samen met de ontwikkelaar gedaan, omdat deze meer kennis van Java heeft. Als de overzetting voltooid is wordt dit aan Software Control gemeld. 5. Software Control zorgt ervoor dat de betrokken partijen worden geïnformeerd over de overzetting.
Postconditie	De software is succesvol overgezet naar de EXP-omgeving en de bijbehorende administratieve taken zijn afgehandeld.

Use case 2.6: Stappen die worden uitgevoerd voor de overzetting van de GAT- naar de EXP-omgeving.

2.2.4 Exploitatie

In de EXP-omgeving moet worden aangetoond dat de software gereed is voor productie. De technische invulling van de omgeving moet zoveel mogelijk overeenkomen met de productieomgeving. De afdeling I&E is verantwoordelijk voor het technisch beheer. Er kan gebruik worden gemaakt van afgeleide omgevingen,

zodat meerdere tests gelijktijdig kunnen worden uitgevoerd. Onderlinge afspraken bepalen wie van welke omgeving gebruik maakt.

Productieacceptatietest

Door de technisch beheerders wordt een productieacceptatietest uitgevoerd. In de praktijk gebeurt dit in samenwerking met de functioneel beheerder, terwijl deze niet bij de afdeling I&E hoort. Het doel van deze test is nagaan of de opgeleverde software kan worden vrijgegeven voor productie en dat alle technische beheeractiviteiten naar behoren kunnen worden uitgevoerd. De test richt zich op het gehele systeem inclusief de bijbehorende infrastructuur. Hierbij worden verschillende onderdelen onderscheiden, waaronder:

- **Performancetest:** Heeft als doel te controleren of de systemen stabiel en snel genoeg blijven draaien wanneer de software grote hoeveelheden data aan het verwerken is. Hierbij worden verschillende tools gebruikt, waaronder JProfiler en JRat. De testcases worden geschreven door de afdeling I&E, die ook verantwoordelijk is voor de uitvoering van de test.
- **Hacktest:** Heeft als doel de software te controleren op beveiligingsrisico's en fouten. Er is een checklist beschikbaar met onderdelen die getest moeten worden. Per geval kunnen hier nog extra tests aan worden toegevoegd.

Naast bovenstaande tests wordt bepaald of de software voldoet aan de overige acceptatie-eisen van I&E, zoals de beheersbaarheid, monitoring en het technisch functioneren. De resultaten worden verwerkt in een acceptatiedossier dat na afloop wordt opgeleverd. Aan de hand van dit document wordt besloten of de software in productie wordt genomen. De cluster Beveiliging van I&E kan buiten dit besluit om aangeven dat een versie van de software in verband met de veiligheid niet naar productie mag.

Overzetten

Als er toestemming is om naar productie te gaan kan de software worden overgezet. De stappen die hiervoor nodig zijn staan beschreven in use case 2.7.

	Overzetten EXP → PRD
Omschrijving	Het overzetten van de software van de EXP- naar de PRD-omgeving.
Actor	De voor deze test verantwoordelijk technisch beheerder.
Preconditie	• De juiste versie van de software staat in de EXP-omgeving. • Er is een akkoord voor het in productie nemen van de software. • Beveiliging is akkoord met het in productie nemen van de software.

	Overzetten EXP → PRD
Basisverloop	1. De voor deze test verantwoordelijk technisch beheerder meldt aan Software Control dat de software kan worden overgezet naar de PRD-omgeving. 2. Software Control voegt de overzetgegevens aan de vrachtbrief toe en stuurt deze (inclusief Turnoverformulier) naar de uitvoerende partijen. 3. Procesmanagement stuurt een goedkeuring voor de overzetting naar de uitvoerende partijen. 4. PROBE voert de automatische databaseoverzetting uit via Turnover. Daarna worden de eventuele handmatige acties op basis van de vrachtbrief uitgevoerd. Als de overzetting voltooid is wordt dit aan Software Control gemeld. 5. De NT-Beheerder voert de Java-overzetting uit door het volgen van het stappenplan uit de vrachtbrief. In de praktijk wordt dit vaak samen met de ontwikkelaar gedaan, omdat deze meer kennis van Java heeft. Als de overzetting voltooid is wordt dit aan Software Control gemeld. 6. Operationele beveiliging zorgt voor de juiste autorisaties. 7. Software Control zorgt ervoor dat de betrokken partijen worden geïnformeerd over de overzetting.
Postconditie	De software is succesvol overgezet naar de PRD-omgeving en de bijbehorende administratieve taken zijn afgehandeld.

Use case 2.7: Stappen die worden uitgevoerd voor de overzetting van de EXP- naar de PRD-omgeving.

2.2.5 Productie

In de productieomgeving bevinden zich de definitieve versies van de software. Dit zijn de systemen die door de eindgebruikers gebruikt worden. Elke applicatie heeft een eigen businesseigenaar. I&E is verantwoordelijk voor het beheer van de resources. De productieomgeving is geen testomgeving en valt buiten de scope van dit onderzoek.

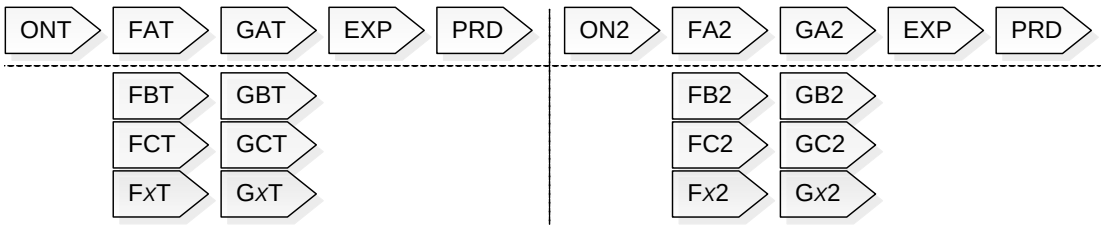
2.3 Aanvullingen op de ontwikkelstraat

Naast de in de vorige paragraaf beschreven omgevingen zijn er nog aanvullende varianten. De afgeleide omgeving, de VT-omgeving, de PT (proeftuin)-omgeving en het LAB zijn in deze paragraaf beschreven.

2.3.1 Afgeleide omgevingen

Om meerdere testers gelijktijdig te kunnen laten testen zijn er van sommige applicaties kopieën beschikbaar. Dit worden afgeleide omgevingen genoemd. De

versie van de programmatuur en de structuur van de database zijn gelijk aan die van de hoofdomgeving, maar iedere afgeleide omgeving heeft zijn eigen inhoud van de database. Figuur 2.3 bevat een voorbeeld van afgeleide omgevingen. Per applicatie bestaat altijd maar één productieomgeving, maar er kunnen meerdere testomgevingen zijn. Momenteel zijn er naast de volledig redundante productieomgeving meer dan 4000 (afgeleide) testomgevingen.



Figuur 2.3: Een hoofdstraat (links) en een parallelle straat (rechts) en de afgeleide omgevingen van beide straten. Hierdoor kan gelijktijdig getest worden.

2.3.2 Veldtest en proeftuin

Binnen de GAT-omgeving kunnen zich twee speciale afgeleide omgevingen bevinden, namelijk de VT- en PT-omgeving. Deze omgevingen zijn geen verplicht onderdeel van de ontwikkelstraat en ze zijn bedoeld om te kunnen testen met onder andere de systemen van onderwijsinstanties.

De omgevingen hebben de volgende doeleinden:

- **VT-omgeving**: Een afgeleide GAT-omgeving waarbij communicatie naar buiten toe mogelijk is. Op deze manier kan een nieuwe versie met externe partijen getest worden.
- **PT-omgeving**: Een omgeving met een productieversie van de programmatuur en daarmee in feite een afgeleide van de GA8 (de GAT-omgeving uit de 8-straat). De PT is bedoeld voor externe partijen die op deze manier hun aansluiting op de DUO-systemen kunnen testen.

2.3.3 Het LAB

Vernieuwingen in de infrastructuur vereisen een andere manier van testen. Hiervoor is de infrastructuur opgesplitst in twee gescheiden omgevingen: het LAB en de productieomgeving. Met de term “omgeving” wordt gescheiden infrastructuurcomponenten bedoeld. In de LAB-omgeving worden nieuwe infrastructuurcomponenten, nieuwe systeemsoftware en standaard softwarepakketten getest.

De omgevingen voor ontwikkelen en testen van software bevinden zich in de infrastructuur van de productieomgeving. Wanneer er bij het testen van maatwerksoftware risico's zijn die de infrastructuur van de productieomgeving zouden kunnen beïnvloeden, wordt er van het LAB gebruik gemaakt.

2.4 Tooling

In deze paragraaf zijn de tools beschreven die momenteel door DUO worden gebruikt bij het ontwikkelen en testen van Java-software. Het betreft hier zowel tools waarmee daadwerkelijk getest wordt als ondersteunende tools, zoals de registratie van bevindingen. De tools staan beschreven in tabel 2.1. In bijlage D staan van enkele tools aanvullende gegevens vermeld.

Tool	Omschrijving
ANT	Automatiseert het buildproces van een Java-project. In een configuratiebestand kunnen taken worden gespecificeerd voor het compileren, assembleren, testen en starten van Java-software.
BareTail	Wordt gebruikt voor het weergeven van logbestanden. De belangrijkste functie is het automatisch verversen van de bestanden.
CruiseControl	Gebruikt ANT voor het automatisch builden van Java-projecten. Via een webinterface kan de status van de huidige en vorige builds worden opgevraagd.
CVS	CVS (Concurrent Versioning System) is een versiebeheersysteem dat bij DUO wordt gebruikt voor Java-code. Centraal in de repository worden versies van bestanden bijgehouden, terwijl ontwikkelaars met een lokale kopie werken.
ITSM	Wordt gebruikt voor de administratie van de IT-dienstverlening bij DUO in de CMDB (Configuration Management Database). Dit is de gegevensverzameling waarin informatie met betrekking tot de CI's (Configuration Items) wordt vastgelegd en beheerd. Op deze manier zijn de relaties tussen verschillende onderdelen vastgelegd. Een CI is een component dat deel uitmaakt van of direct gerelateerd is aan de IT-infrastructuur. Een samengestelde groep van items kan zelf ook weer als een item worden gezien. Daarnaast wordt ITSM (met name door I&E) gebruikt voor de administratie en (micro)planning voor de uitvoering van RFC's. De tool werkt ondersteunend voor de IT-servicemanagement en ICT-Demand processen. ↳ Zie ook: bijlage D.1
JAMon (Java Application Monitor)	Een opensource framework om metingen vanuit de Javacode uit te voeren. Hierdoor zijn onderdelen van applicaties te monitoren. De resultaten kunnen met behulp van het opensource JARep framework eenvoudig worden gefiltert en overzichtelijk (met behulp van grafieken) worden weergegeven.
JARep	Een opensource framework voor het grafisch weergegeven van performance data (die verkregen is via bijvoorbeeld JAMon) over een bepaalde tijd.

Tool	Omschrijving
JCC (Java Control Center)	Een applicatie waarmee het mogelijk is bepaalde taken (zoals starten, stoppen, pauzeren en de status opvragen) voor Java-software uit te voeren. Om gebruik te maken van deze functionaliteit moet een connector worden geconfigureerd en in de software worden geïntegreerd.
Jira	Wordt als pilot gebruikt en werkt ondersteunend voor het ontwikkelproces. Een belangrijk onderdeel is het vastleggen en afhandelen van testbevindingen. Deze tool gaat mogelijk ClearQuest vervangen.
JMeter	Een tool waarmee functionele en performancetests kunnen worden uitgevoerd. De tool meet de tijd tussen het versturen van een request en het ontvangen van het antwoord.
JProfiler	Een profilingtool die bij DUO vaak in combinatie met de WebSphere applicatieserver wordt gebruikt voor het uitvoeren van performancetests.
JRat	Een eenvoudige profilingtool waarmee wordt aangegeven waar de processortijd aan besteed wordt en wat het totale geheugengebruik door de tijd heen is.
LSD (Lokaal Snel Draaien)	Een door DUO ontwikkelde invoegtoepassing voor Excel die geautomatiseerd testen lokaal (vanaf de eigen PC) mogelijk maakt. Gegevens kunnen automatisch en daardoor snel ingevoerd worden op legacysystemen, HTML-pagina's en XML-berichten (voor services). Met behulp van actiewoorden kan een complete test worden gemaakt. Het kan ook worden gebruikt om een uitgangspositie voor een test klaar te zetten. Daarnaast is het mogelijk het testdossier aan een draai-PC aan te bieden. Dit heeft als voordeel dat de eigen PC voor andere taken kan worden gebruikt. ↳ Zie ook: bijlage D.2
MagicDraw	Een UML-modelleringstool waar met behulp van de Teamworkserver in teamverband mee kan worden gewerkt. Bij DUO wordt onder andere het service- en berichtenboek hier in bijgehouden. MagicDraw wordt gebruikt voor het automatisch genereren van WSDL's. Dit zorgt ervoor dat de documentatie klopt met de werkelijkheid. Verder zijn er exportmogelijkheden voor het genereren van documentatie. ↳ Zie ook: bijlage D.3
MQJExplorer	Een grafische managementtool voor WebSphere MQ. Een belangrijke functie is de mogelijkheid om MQ-berichten te lezen van of berichten toe te voegen aan een queue.
Rational ClearQuest	Tool die wordt gebruikt voor het registreren van bevindingen. Er wordt alleen gebruik gemaakt van de Windows client. Iedere bevinding is voorzien van een status, deze status kan worden veranderd door het uitvoeren van een actie. Op deze manier wordt het hele traject dat de bevinding heeft afgelegd geregistreerd. ClearQuest en ITSM zijn niet automatisch gekoppeld. Om toch een link te kunnen leggen worden de RFC-ID, omschrijving en status uit ITSM overgenomen in ClearQuest. Dit betreft een handmatige actie. ↳ Zie ook: bijlage D.4

Tool	Omschrijving
Selenium	Een record en playback tool waarmee geautomatiseerd schermen kunnen worden getest, inclusief het controleren van de proefgevallen.
SoapUI	Een opensource tool voor het testen van webservices op een SOA. Wordt bij DUO gebruikt voor het maken, versturen en ontvangen van XML-berichten. Een hele serie berichten kan als testsuite worden opgeslagen en worden uitgevoerd. SoapUI wordt tevens gebruikt voor het maken van mockservices.
Sonar	Een opensource continuous inspection tool voor het bewaken van de kwaliteit van software. Source code wordt geanalyseerd aan de hand van ingestelde regels en tests. De resultaten zijn via een webinterface te bekijken.
Squirrel	Een tool waarmee databases te benaderen zijn. De inhoud van de tabellen kan worden bekeken en er kunnen mutaties worden uitgevoerd.
Teamwork-server	Serversoftware die wordt gebruikt voor de centrale opslag van MagicDraw projecten.
TestZipper	Een door DUO ontwikkelde tool die wordt gebruikt voor het versiebeheer van testdossiers op de netwerkshare. Doordat de tool een deel van de acties automatiseert zijn er minder handmatige acties nodig.
TFAT	Een door CMG ontwikkelde tool voor het automatisch genereren van beslistabellen en vereenvoudigde beslistabellen. De laatste versie van de tool is uitgebracht in 2003 en er komen geen nieuwe versies meer uit. De resultaten uit TFAT kunnen worden geïmporteerd in het testdossier.
Turnover	Wordt gebruikt voor het automatisch overzetten van legacysoftware en DB2-databases op het AS/400 platform.
VTAnalyzer	Wordt gebruikt om vastgelegde bevindingen van een gekozen release uit ClearQuest te halen en daar bewerkingen op los te laten. Met behulp van grafieken kan worden weergegeven hoe het testtraject ervoor staat.
WinMerge	Wordt gebruikt voor het vergelijken van bestanden en het resultaat van de vergelijking op te slaan in een DIFF-bestand.
WSAdmin	Tool van IBM om administratortaken op een WebSphere applicatieserver via de console uit te kunnen voeren. Deze tool wordt ook gebruikt bij het uitvoeren van Jython-scripts.

Tabel 2.1: Overzicht van de huidige tools. Bij enkele tools wordt verwezen naar een bijlage voor aanvullende informatie.

2.5 Problemen en verbeterpunten

In de vorige paragrafen is beschreven hoe de huidige situatie eruitziet, met daarbij de verschillende omgevingen, procedures en gebruikte tools. De afgelopen jaren is de manier van ontwikkelen veranderd. Er wordt gebruik gemaakt van een andere ontwikkelmethodiek (Scrum), nieuwe technieken en er ontstaan meer afhankelijkheden tussen de verschillende softwaresystemen. Veranderingen als deze zorgen ervoor dat de huidige manier van werken problemen met zich meebrengt.

De geconstateerde problemen zijn beschreven in tabel 2.2. Deze probleempunten zijn tijdens gesprekken met de medewerkers naar voren gekomen. Ieder probleem is voorzien van een uniek ID, zodat hier naar verwezen kan worden.

ID	Onderwerp	Omschrijving
	Ontwikkeling	
PB-01	Ontwikkel-methodiek	De omgevingen zijn gebaseerd op de watervalmethodiek, terwijl er steeds meer ontwikkeld wordt volgens de Scrum methodiek. Hierdoor wordt op een andere manier getest, zijn er vaker overzettingen en zijn omgevingen al in een eerder stadium van het project nodig.
PB-02	Servers	In de ONT- en FAT-omgeving staat een aantal servers te vol met applicaties, doordat bouwers zelf applicaties op de server zetten. Er is geen duidelijk inzicht welke applicaties nog worden gebruikt en wat kan worden verwijderd. Dit zorgt ervoor dat testen vertraging oplopen vanwege problemen met de servers.
	Afdelingen	
PB-03	Opsplitsing	Door de verschillende afdelingen, verantwoordelijkheden en omgevingen wordt er te veel naar het eigen deel gekeken in plaats van naar het geheel. Zelfs bij het Softwarehuis is een scheiding te zien. Zowel de ontwikkelaars als de testers hebben hun eigen verzameling tools en de link hiertussen mist.
PB-04	Locaties	Er is op dit moment een fysieke scheiding (ander gebouw) tussen het Softwarehuis en I&E. Nu verloopt het contact vooral digitaal, wat de samenwerking niet ten goede komt.
	Omgevingen	
PB-05	Overzicht	Er is weinig overzicht wie van welke omgevingen gebruik maakt. Hierdoor werken ontwikkelaars en testers elkaar soms tegen.
PB-06	Ondersteuning	De testomgevingen worden ondersteund door de mensen die de testtools beheren, terwijl deze verantwoordelijkheid eigenlijk bij de technisch beheerders zou moeten liggen.
PB-07	Beheer	Bij de introductie van Java heeft het Softwarehuis de taak van het omgevingenbeheer opgepakt voor de ONT- en FAT-omgeving. De GAT-, EXP- en PRD-omgeving worden wel door I&E beheerd. Doordat het technisch beheer door twee partijen wordt uitgevoerd mist het overzicht en ontstaan er soms te grote verschillen tussen de omgevingen. Dit zorgt voor problemen met overzettingen.
PB-08	Configuratie	De EXP-omgeving is altijd productielike, waardoor bij de overzetting naar deze omgeving de software soms niet meer juist werkt vanwege de verschillen in de configuratie, zoals andere versies van frameworks, netwerkinstellingen en de aanwezigheid van firewalls.

ID	Onderwerp	Omschrijving
PB-09	Verantwoording	Geen enkele partij voelt zich echt verantwoordelijk voor het technisch beheer van de testomgevingen. Er zijn geen duidelijke afspraken gemaakt over de verantwoordelijkheden.
	Testen	
PB-10	Wijzigingen	Bij de legacysoftware worden alleen de wijzigingen overgezet. Bij Java kan dit niet langer en wordt altijd de volledige applicatie overgezet. Hierdoor kan aan de overzetting zelf niet meer duidelijk worden gezien wat er is veranderd, waardoor vaker een regressietest moet worden uitgevoerd.
PB-11	Logbestanden	De ontwikkelaar heeft geen toegang tot de juiste logbestanden als er in de GAT-omgeving een bevinding wordt gedaan. Het opvragen van logbestanden bij NT-Beheer kost veel tijd.
PB-12	Exploitatie	De tests in de EXP-omgeving worden vooral door ICT-Demand uitgevoerd, terwijl de verantwoordelijkheid voor deze tests bij I&E ligt. Dit komt omdat NT-Beheer over onvoldoende kennis beschikt.
PB-13	Keten	Applicaties maken tegenwoordig gebruik van vele services, waarbij deze services ook weer van een service gebruik kunnen maken. Indien er iets fout gaat is het lastig te achterhalen waar het probleem zich in de keten voordoet.
	Overzettingen	
PB-14	Handmatige acties	Java-overzettingen bestaan momenteel uit veel handmatige acties. Dit maakt het overzetten ingewikkelder en daardoor is de kans groter dat er fouten worden gemaakt.
PB-15	Kennis	Vrachtbrieven zijn niet altijd gedetailleerd genoeg, waardoor de software na het uitvoeren van het stappenplan niet werkt. NT-Beheer heeft te weinig kennis van Java om deze problemen zelf op te lossen. Aan de invulling van de vrachtbrief (onvolledige stappen) is vaak al te zien of er problemen gaan optreden.

Tabel 2.2: Geconstateerde problemen met de huidige situatie. Elk probleem is voorzien van een ID, zodat er naar verwezen kan worden.

Naast de probleempunten zijn er ook een aantal verbeterpunten geconstateerd. Dit zijn punten die door de medewerkers niet direct als probleem worden gezien, maar in de toekomst wel verbeterd kunnen worden. De verbeterpunten zijn beschreven in tabel 2.3.

ID	Onderwerp	Omschrijving
VP-01	E-mail	Documenten en statusupdates worden vaak via e-mail verzonden en niet in een centraal systeem bijgehouden.

ID	Onderwerp	Omschrijving
VP-02	Testdata	Binnen de ONT- en de FAT-omgeving wordt vanwege privacy-regels niet gewerkt met een kopie van de productiedata. Het anonimiseren van productiedata geeft de mogelijkheid om ook in deze omgevingen met productielike data te werken. Door in de verschillende omgevingen met dezelfde soort data te werken komen data-gerelateerde problemen eerder naar voren en kan een ontwikkelaar een bevinding eenvoudiger reproduceren.
VP-03	Koppeling	Er is geen automatische koppeling tussen ITSM en ClearQuest. Aan de hand van het ID moet zelf in de andere software worden gezocht en een deel van de gegevens moet handmatig worden overgenomen. Een centraal systeem met een duidelijke koppeling zal dit proces eenvoudiger en overzichtelijker maken.
VP-04	Excel	Er wordt veel gewerkt met Excel, waarbij gebruik wordt gemaakt van zelfontwikkelde invoegtoepassingen. Hierbij zijn de grenzen van wat mogelijk is met Excel opgezocht. Een update naar een nieuwe versie van Excel gaat mogelijk voor problemen zorgen. Tevens wordt er gewerkt met losse Excel-bestanden, zie hiervoor VP-05.
VP-05	Documentbeheer	Door het gebruik van Excel (zie VP-04) en e-mail wordt veel gewerkt met losse bestanden voor documentatie en testdossiers. Deze bestanden bevinden zich op netwerkshares waarbij de mappenstructuur ervoor moet zorgen dat het geheel te beheren blijft. Een centraal systeem met duidelijk versiebeheer zal dit proces vereenvoudigen. Excel zal dan (stapsgewijs) kunnen worden uitgefaseerd, waarbij de gegevens worden overgenomen in een centraal systeem.

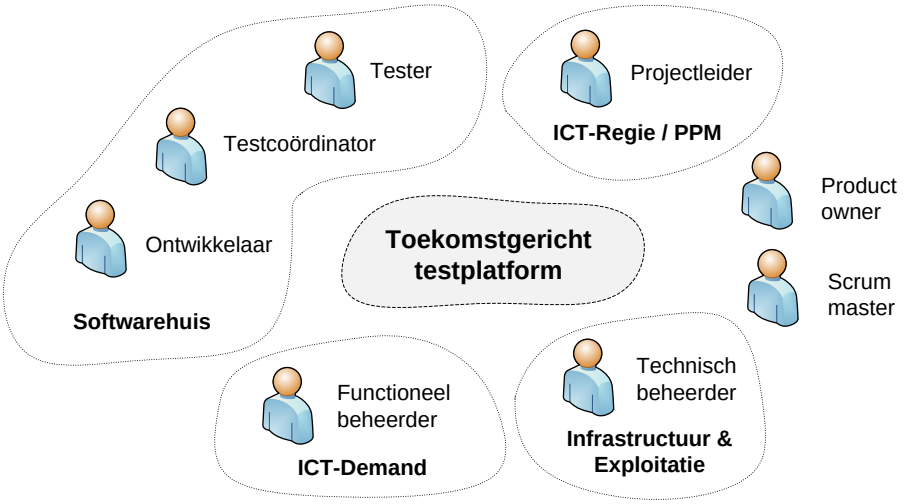
Tabel 2.3: Geconstateerde verbeterpunten met betrekking tot de huidige situatie. Elk verbeterpunt is voorzien van een ID, zodat er naar verwezen kan worden.

3 Gewenste situatie

Om vast te stellen wat er wordt verwacht van een toekomstgericht testplatform is het van belang de gewenste situatie te onderzoeken. Met behulp van requirements engineering zijn de stakeholders, user stories, functionele requirements en niet-functionele requirements in kaart gebracht. De inhoud is tot stand gekomen met de informatie uit de gesprekken, door rekening te houden met de concept ICT-principes (zie bijlage E) en door eigen inbreng van de afstudeerders. Vervolgens zijn controlegesprekken gevoerd om de inhoud aan te vullen en te verbeteren.

3.1 Stakeholders

Stakeholders zijn personen of afdelingen binnen een organisatie die een belang hebben bij een project [8]. Figuur 3.1 geeft een overzicht van de stakeholders en bijbehorende afdelingen. In tabel 3.1 staan de belangen vermeld.



Figuur 3.1: Overzicht van de stakeholders en afdelingen. De afdeling van de Scrum master en product owner kan per project verschillen.

ID	Stakeholder	Belangen
SH-01	Ontwikkelaar	Minder tijd besteden aan het oplossen van problemen met overzettingen tussen omgevingen. Hierdoor houdt een ontwikkelaar meer tijd over voor zijn kerntaken.
SH-02	Tester	Moeiteloos kunnen testen van software, zonder het testplatform zelf als obstakel te hebben. Als een test mislukt, voldoende informatie kunnen verzamelen voor een duidelijke bevinding.

ID	Stakeholder	Belangen
SH-03	Testcoördinator	Een hogere kwaliteit van testen leveren, sneller een test kunnen inplannen en inzicht krijgen in het gebruik van testomgevingen.
SH-04	Technisch beheerder	Een testplatform die onder het beheer van één afdeling valt en eenvoudig en snel te beheren is.
SH-05	Functioneel beheerder	Probleemloos een gebruikersacceptatietest kunnen uitvoeren. De belangen van SH-02 zijn hierbij ook van toepassing.
SH-06	Scrum master	Het testplatform mag geen risico vormen voor de planning van een sprint. Beschikbaarheid en stabiliteit is hierbij belangrijk.
SH-07	Product owner	Tijdens de demo van een sprint een werkend product kunnen zien. Hier is een stabiele testomgeving voor vereist.
SH-08	Projectleider	Vertraging van projecten voorkomen met de juiste voorzieningen.

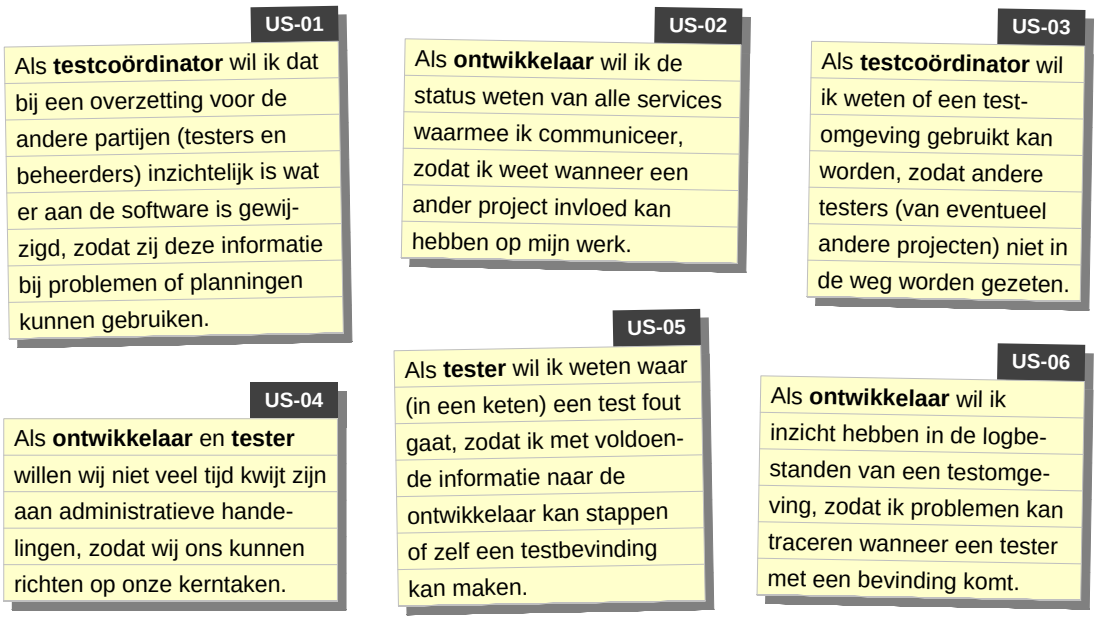
Tabel 3.1: De belangen van de stakeholders voor dit onderzoek.

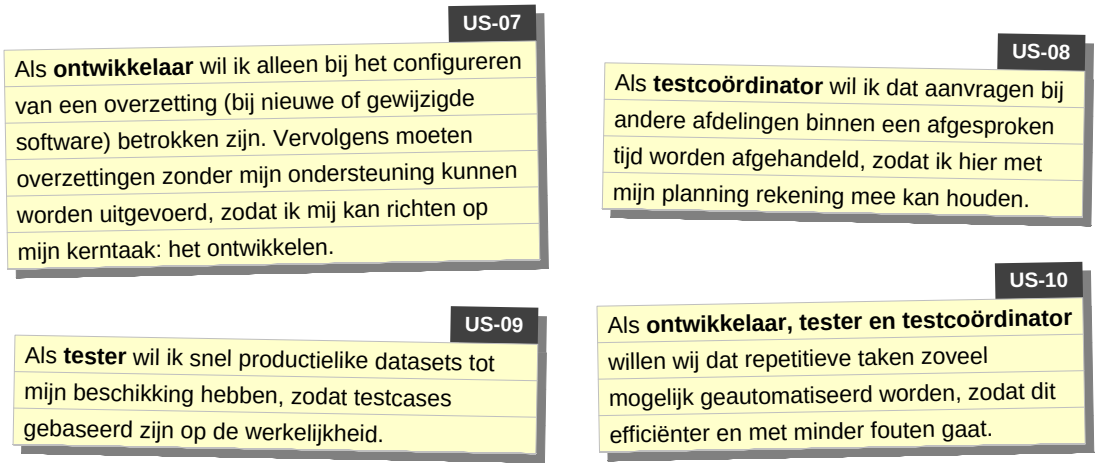
3.2 User stories

Van elke stakeholder zijn de belangrijkste user stories verzameld. Een user story beschrijft in een aantal zinnen wat de gebruiker van een systeem nodig heeft om zijn taken te kunnen vervullen. Ze vormen een belangrijke input voor de requirements van het toekomstgerichte testplatform.

3.2.1 Softwarehuis

Het Softwarehuis levert softwareontwikkelaars, testcoördinatoren en testers aan projecten. De user stories van het Softwarehuis staan in figuur 3.2.

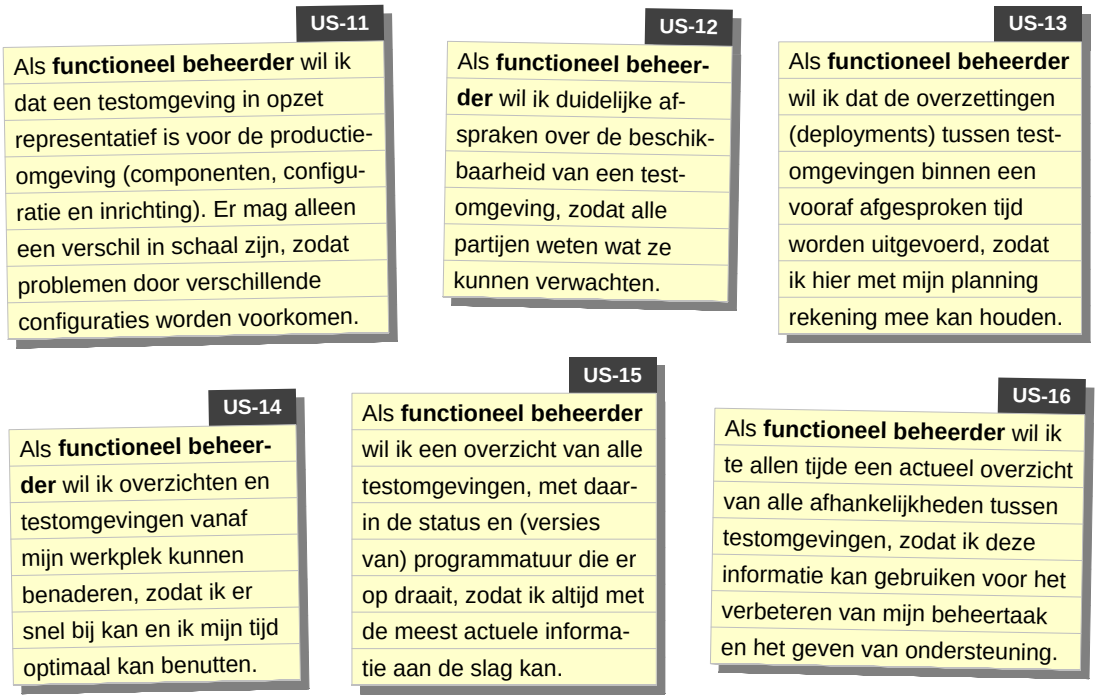




Figuur 3.2: Overzicht van user stories van de ontwikkelaars, testers en testcoördinatoren van het Softwarehuis.

3.2.2 ICT-Demand

ICT-Demand is verantwoordelijk voor de acceptatie van nieuwe en gewijzigde programmatuur en geeft ondersteuning aan de eindgebruiker. De user stories van de afdeling ICT-Demand staan in figuur 3.3.



Figuur 3.3: Overzicht van user stories van de functionele beheerders van de afdeling ICT-Demand.

3.2.3 Infrastructuur & Exploitatie

I&E beheert de technische infrastructuur en de implementatie van de systemen hierop. De user stories van de afdeling I&E staan in figuur 3.4.

Figuur 3.4: Overzicht van user stories van de technische beheerders van de afdeling Infrastructuur & Exploitatie.

3.2.4 ICT-Regie, PPM en Scrum projectrollen

Bij de Scrum ontwikkelmethodiek wordt gewerkt met verschillende rollen. Deze rollen kunnen afhankelijk van het project door verschillende afdelingen worden vervult. Projectleiding wordt geleverd door de afdelingen ICT-Regie en PPM. De user stories met betrekking tot projectleiding en Scrum rollen staan in figuur 3.5.

Figuur 3.5: Overzicht van user stories van de afdelingen ICT-Regie en PPM en de projectrollen van de Scrum ontwikkelmethodiek.

3.3 Key drivers

De belangrijkste requirements voor een systeem worden key drivers genoemd. De beslissingen die een grote invloed hebben op de architectuur van het testplatform (design decisions) moeten sterk rekening houden met deze key drivers. De vier key drivers voor het toekomstgerichte testplatform staan in tabel 3.2 en hebben een duidelijk raakvlak met de user stories uit paragraaf 3.2.

ID	Key driver	Omschrijving
KD-01	Overzicht	Actueel overzicht hebben van de testomgevingen, resourcegebruik, aanwezige programmatuur, versies van de programmatuur en de relaties die hiertussen bestaan.
KD-02	Automatisering	Geautomatiseerd testomgevingen aan kunnen maken en overzettingen tussen omgevingen uit kunnen voeren.
KD-03	Beschikbaarheid	Het testplatform moet een hoog beschikbaarheidspercentage hebben. Projecten mogen geen vertraging oplopen door testomgevingen die niet beschikbaar zijn.
KD-04	Gebruiksvriendelijkheid	De tooling behorende bij het testplatform moet makkelijk in gebruik zijn voor de gebruikers.

Tabel 3.2: De vier key drivers voor het toekomstgerichte testplatform.

3.4 Requirements

Met behulp van de user stories, belangen van de stakeholders en de key drivers zijn de wensen voor een toekomstgericht testplatform vastgelegd. In deze paragraaf zijn de wensen vertaald naar functionele en niet-functionele requirements [9]. Om backwards traceability te garanderen zijn de requirements gekoppeld aan de probleempunten, verbeterpunten, user stories en key drivers. Ook bevat elke requirement een uniek ID en een prioriteit: L (laag), G (gemiddeld) of H (hoog).

Uit de key drivers kan worden afgeleid dat er een softwaresysteem nodig zal zijn. Voor dit systeem is de werktitel TPM (Testplatform Manager) geïntroduceerd. Deze werktitel wordt in de rest van het document gebruikt.

Definitie van TPM

TPM staat voor Testplatform Manager en is de werktitel voor het softwaresysteem van het toekomstgerichte testplatform.

3.4.1 Functionele requirements

Een functionele requirement beschrijft een functie die een systeem dient te vervullen. Tabel 3.3 bevat een overzicht van de functionele requirements voor het toekomstgerichte testplatform.

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-01	**Het beheren van en inzicht krijgen in het testplatform moet vanuit één softwaresysteem gebeuren.** *Motivering:* Eén systeem is sneller en overzichtelijker in gebruik dan een fragmentatie van systemen.	PB-03 VP-03 VP-04 VP-05	KD-01	H
FR-02	**TPM moet een webapplicatie zijn die vanaf de werkplek van elke stakeholder te benaderen is.** Via het intranet kan de webapplicatie op bijvoorbeeld `http://tpm/` beschikbaar worden gemaakt. *Motivering:* Een webapplicatie is eenvoudig te benaderen en vereist geen installatie van software op de werkplekken van de stakeholders.	VP-05	KD-04 US-14	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-03	**In TPM moeten projecten kunnen worden aangemaakt.** Een project bevat minimaal de volgende gegevens: a. een uniek ID b. een naam c. een omschrijving. *Motivering:* Het is voor het overzicht en de structuur belangrijk dat projecten kunnen worden toegevoegd. Het sluit ook beter aan bij de Scrum ontwikkelmethodiek.	PB-05 VP-03	–	H
FR-04	**Voor een project moet via TPM een testomgeving kunnen worden aangevraagd.** Een testomgeving moet zijn voorzien van de volgende gegevens: a. een uniek ID b. het project-ID c. de aanvrager d. de servicecategorie (zie FR-05) e. de voorzieningen (zie FR-06) f. een naam (optioneel). *Motivering:* Afhankelijk van het project kunnen de benodigde testomgevingen verschillen, deze moeten daarom aangevraagd worden.	PB-14 VP-03	KD-02 US-29	H
FR-05	**Bij een testomgeving moet kunnen worden aangegeven welke service-eisen van toepassing zijn.** Met een categorie moet worden aangegeven welke service-eisen (zoals beschikbaarheid) voor de testomgeving van toepassing zijn. *Motivering:* Niet elke testomgeving zal aan dezelfde eisen moeten voldoen. Zo zal een omgeving voor demonstraties een hogere beschikbaarheid moeten hebben.	PB-09	KD-03 US-08 US-12 US-24 US-30	H
FR-06	**Testomgevingen moeten met een bepaalde set van voorzieningen kunnen worden aangemaakt.** Voorzieningen zijn voorgeïnstalleerde combinatie van een besturingssysteem, tooling en frameworks. *Motivering:* Door direct bij de aanvraag aan te geven welke voorzieningen nodig zijn, kan dit sneller worden gerealiseerd. Tevens heeft de technisch beheerder getest dat deze voorzieningen juist werken.	PB-14	KD-02 US-19	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-07	**Het moet duidelijk zijn met welke voorzieningen (FR-06) een testomgeving kan worden aangemaakt.** *Motivering:* Voor de gebruiker moet bij een aanvraag duidelijk zijn uit welke voorzieningen kan worden gekozen.	PB-14	KD-02	H
FR-08	**In TPM moet een aanvraag voor het verwijderen van een testomgeving kunnen worden gedaan.** Als een testomgeving binnen een project niet langer noodzakelijk is, moet ervoor kunnen worden gekozen om deze te laten verwijderen. *Motivering:* Het is belangrijk dat er niet onnodig veel testomgevingen zijn en dat daarmee resources onnodig lang gereserveerd blijven.	PB-02 PB-14	US-21	H
FR-09	**In TPM moeten overzettingen tussen omgevingen kunnen worden aangevraagd.** Bij een aanvraag moet kunnen worden aangegeven wat de reden van de overzetting is. *Motivering:* Dit moet een aanvraag zijn omdat de overzetting niet in alle gevallen direct uitgevoerd mag worden.	VP-01	KD-02	H
FR-10	**Alle acties die in TPM worden uitgevoerd (zowel gestart door een gebruiker als geautomatiseerde taken) moeten worden gelogd.** Onder acties worden de volgende handelingen verstaan: • het toevoegen van een entiteit (bijvoorbeeld: testomgeving, relatie of versienummer) • het wijzigen van een entiteit • het verwijderen van een entiteit. De volgende gegevens moeten worden gelogd: a. een omschrijving van de actie die is uitgevoerd b. de datum en het tijdstip waarop de actie is uitgevoerd c. de persoon of het systeem die de actie heeft uitgevoerd. *Motivering:* Met logging kunnen acties uit het verleden worden onderzocht. Dit kan helpen bij het oplossen van problemen op het testplatform.	PB-11 PB-13	KD-01 US-06 US-22	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-11	**TPM moet een actueel overzicht van het testplatform aanbieden.** De volgende gegevens moeten op een overzichtelijke manier kunnen worden getoond: a. Alle testomgevingen die op het moment aanwezig zijn. b. Het project waar de testomgeving onderdeel van is. c. De status van elke testomgeving. Hieruit moet af te leiden zijn of een testomgeving beschikbaar is of dat deze wordt gebruikt voor een test, onderhoud of andere werkzaamheden. d. De Java-programmatuur die op een testomgeving geïnstalleerd is, inclusief versienummering. e. Alle afhankelijkheden die tussen testomgevingen bestaan. Voorbeeld: versie 0.4 van service A op omgeving X maakt gebruik van versie 0.3 van service B op omgeving Y. f. De versie van de functionele documentatie waar de software in de omgeving op gebaseerd is. g. Overzicht van het resourcegebruik per testomgeving (CPU, geheugen, hardeschijfruimte, hardeschijf I/O). *Motivering:* Het bieden van overzicht is een key driver en een belangrijke functionaliteit die TPM moet bieden.	PB-05 VP-05	KD-01 US-03 US-15 US-16 US-23	H
FR-12	**Het overzetten van programmatuur tussen omgevingen moet geautomatiseerd worden uitgevoerd.** Het aangeven dat een overzetting tussen twee omgevingen gestart moet worden, is de enige menselijke handeling die nodig moet zijn. *Motivering:* Handmatige overzettingen kosten veel tijd. Bovendien is de kans dat er bij een menselijke handeling een fout gemaakt wordt sterk aanwezig.	PB-14	KD-02 US-04 US-07 US-10 US-13 US-17	H
FR-13	**Afhankelijkheden tussen softwarecomponenten en omgevingen moeten zoveel mogelijk automatisch worden gedetecteerd en in TPM worden vastgelegd.** *Motivering:* Door koppelingen ontstaan steeds complexere afhankelijkheden. Het is belangrijk dat deze afhankelijkheden vastgelegd zijn, zodat de systemen en de gebruikers deze informatie kunnen gebruiken.	PB-05 PB-13 PB-14	KD-01 KD-02 US-02 US-16 US-31	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-14	**TPM moet acties behorende bij één of meerdere projecten overzichtelijk op chronologische volgorde kunnen weergeven (een tijdlijn).** Er moet eenvoudig door de tijd kunnen worden genavigeerd, waardoor het verloop van één of meerdere projecten duidelijk inzichtelijk is. *Motivering:* Het verloop van een project kan inzichten geven waarom bepaalde situaties zijn ontstaan.	PB-05 PB-13 VP-05	KD-01 KD-04 US-06 US-22	H
FR-15	**Het resourcegebruik van een testomgeving moet worden bijgehouden.** De volgende informatie moet beschikbaar zijn: a. CPU-gebruik b. geheugenruimte c. hardeschijfruimte. *Motivering:* Het is belangrijk een duidelijk beeld te hebben in hoeverre de gekozen hoeveelheid resources daadwerkelijk worden gebruikt.	PB-02	KD-02 KD-03 US-25	H
FR-16	**TPM moet een melding geven als een omgeving gedurende een ingestelde periode niet gebruikt is.** *Motivering:* Het aantal omgevingen moet beperkt blijven en resources moeten niet onnodig gereserveerd blijven.	PB-02	KD-02 US-21	G
FR-17	**Aan het gebruik van testomgevingen zijn kosten verbonden. Deze kosten moeten bij de aanvraag duidelijk zijn.** De kosten zijn afhankelijk van de hoeveelheid resources en de categorie van de testomgeving. *Motivering:* Resources kosten geld en het gebruik hiervan moet dan ook worden doorberekend aan de projecten. Tevens zorgt deze manier van werken ervoor dat ongebruikte omgevingen eerder worden verwijderd.	PB-09	US-21 US-25	G
FR-18	**Een testomgeving moet automatisch kunnen worden teruggezet naar een "clean-state".** *Motivering:* Indien er problemen zijn met een omgeving moet het mogelijk zijn de omgeving te herstellen door deze terug te zetten naar een "clean-state".	PB-14 VP-02	KD-02 US-10 US-19	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-19	**In TPM moet met rollen kunnen worden bepaald tot welke functionaliteiten een gebruiker toegang heeft.** De volgende rollen moeten beschikbaar zijn: • projectleider • functioneel beheerder • technisch beheerder • ontwikkelaar • testcoördinator • tester. *Motivering:* Het is belangrijk dat een gebruiker alleen de functionaliteiten kan gebruiken die voor zijn taken nodig zijn. Voor bepaalde functies kan de verantwoordelijkheid bij een andere afdeling liggen.	PB-01 PB-07	–	G
FR-20	**Een gebruiker heeft minimaal één rol in TPM, maar moet ook meerdere rollen kunnen hebben.** Beschikbare functionaliteiten zijn gekoppeld aan rollen. Een gebruiker dient hierdoor minimaal één rol te hebben. *Motivering:* Het is onder andere bij Scrum gebruikelijk dat een gebruiker meerdere rollen heeft.	PB-01 PB-07	–	G
FR-21	**Er moeten teamleden aan een project kunnen worden toegevoegd.** Een projectleider moet kunnen bepalen welke teamleden tot een project behoren. *Motivering:* Voor het bepalen van verantwoordelijkheden (zie FR-22) is het noodzakelijk dat duidelijk is welke gebruikers bij een project horen.	PB-01 PB-07	–	G
FR-22	**Per project moeten de teamleden rollen kunnen krijgen.** Een projectleider moet zijn teamleden één of meerdere rollen kunnen toewijzen. *Motivering:* Een gebruiker kan binnen verschillende projecten verschillende verantwoordelijkheden hebben.	PB-01 PB-07	–	G
FR-23	**Er moet een kopie van een testomgeving kunnen worden gemaakt.** Met een kopie van een testomgeving kan een project de beschikking krijgen over een eigen versie van de software en bijbehorende data die voor dat project gebruikt kan worden. *Motivering:* Het is vergelijkbaar met een afgeleide omgeving en in sommige gevallen noodzakelijk voor het testen.	–	KD-02	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-24	**Met behulp van SSO (single sign-on) moet een gebruiker direct ingelogd zijn op TPM.** *Motivering:* Single sign-on voorkomt het onnodig herhaaldelijk moeten inloggen op een systeem. Doordat een gebruiker al is ingelogd op zijn werkplek zijn de gegevens bekend. Opnieuw een handeling van de gebruiker vragen is dus niet nodig.	PB-14	KD-02 KD-04	G
FR-25	**Elke testomgeving moet over een eigen IP-adres beschikken.** Bij het aanmaken van een testomgeving moet deze automatisch een eigen IP-adres toegewezen krijgen. *Motivering:* Veel van de Java-software bij DUO communiceert via het netwerk. Ook moeten beheerders de testomgeving kunnen bereiken via een administratorconsole, waarvoor een eigen IP-adres erg nuttig is.	PB-14	KD-02	G
FR-26	**In TPM moet een aanvraag voor een bepaalde actie (bijvoorbeeld een nieuw project of omgeving) kunnen worden goedgekeurd of afgekeurd.** *Motivering:* Voor sommige aanvragen is goedkeuring van één of meerdere partijen nodig. Door dit centraal via TPM te doen is direct inzichtelijk op welke goedkeuringen nog gewacht wordt.	VP-01	US-18 US-31	G
FR-27	**De status van een aanvraag moet in TPM kunnen worden opgevraagd.** Voorbeelden van een status zijn: • aanvraag ingediend, wacht op goedkeuring • aanvraag goedgekeurd, wacht op een beheerder • in behandeling door een beheerder. *Motivering:* Doordat de status via TPM te bekijken is weet iedereen in welk stadium een aanvraag zich bevindt.	VP-01	KD-01 US-31	G
FR-28	**De logs in TPM moeten overzichtelijk kunnen worden weergegeven.** *Motivering:* Het is belangrijk dat de logs kunnen worden gebruikt voor bijvoorbeeld het oplossen van problemen.	PB-11	KD-01 US-06	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-29	**Bij alle overzichten in TPM moeten filters kunnen worden toepast.** Afhankelijk van de gegevens die worden getoond moet in ieder geval mogelijk zijn om te filteren op: a. testomgeving b. project c. soort actie d. gebruiker e. datum en tijd. *Motivering:* Doordat er veel gegevens kunnen worden getoond is het belangrijk dat de gewenste informatie eenvoudig kan worden gevonden.	PB-11	KD-01 US-06	G
FR-30	**Alle overzichten in TPM moeten kunnen worden gesorteerd.** Afhankelijk van de gegevens die worden getoond moet gesorteerd kunnen worden op: a. testomgeving b. project c. soort actie d. gebruiker e. datum en tijd. *Motivering:* Doordat er veel gegevens kunnen worden getoond is het belangrijk dat de gewenste informatie eenvoudig kan worden gevonden.	PB-11	KD-01 US-06	G
FR-31	**TPM moet voorzien zijn van een berichtenfunctionaliteit.** Via berichten moet de gebruiker van informatie worden voorzien. TPM moet de volgende soorten berichten ondersteunen: a. antwoorden op supporttickets (zie FR-32) b. automatisch gegenereerde berichten met statusupdates van projecten waar de gebruiker aan deelneemt. *Motivering:* Door automatisch berichten te versturen worden afhankelijke partijen altijd op de hoogte gebracht bij veranderingen.	VP-01	KD-02 KD-04 US-02 US-21	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-32	**Voor ondersteuning moet een supportticket kunnen worden aangevraagd via het berichtensysteem.** Bij problemen met of vragen over het testplatform moet een gebruiker een supportticket kunnen aanmaken. *Motivering:* Door te werken met tickets kan worden gegarandeerd dat deze tickets binnen een afgesproken tijd worden behandeld. Daarnaast is de informatie centraal in TPM beschikbaar.	PB-06 PB-09	KD-04 US-04	G
FR-33	**Een supportticket moet een status hebben.** De status laat zien in welk stadium een supportticket zich bevindt. De volgende statussen zijn mogelijk: a. open b. in behandeling c. gesloten. *Motivering:* Een status maakt duidelijk wat afgehandeld is en wat niet. Dit kan gebruikt worden om op te filteren.	PB-06	–	G
FR-34	**Een supportticket moet kunnen worden gekoppeld aan een project.** Door een supportticket aan een project te koppelen is deze inzichtelijk voor de andere teamleden van het project. *Motivering:* Sommige problemen hebben betrekking op een project. Door dit inzichtelijk te maken voor de andere projectleden wordt voorkomen dat er meerdere supporttickets voor hetzelfde probleem worden aangemaakt.	VP-03	KD-01	G
FR-35	**Een project moet een geautomatiseerd bericht ontvangen wanneer een overzetting is aangevraagd of wordt uitgevoerd voor een testomgeving behorende bij het project.** Bij het aanvragen of uitvoeren van een overzetting naar een testomgeving moeten projecten worden geïnformeerd die met deze testomgeving te maken hebben. Via een automatisch bericht worden deze projecten op de hoogte gebracht. *Motivering:* Projecten komen hierdoor niet voor verrassingen te staan en kunnen ingrijpen wanneer dat nodig is.	PB-01 PB-03 PB-07 PB-13 PB-15	KD-02 KD-03 KD-04 US-02	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-36	**Een gebruiker moet een e-mail kunnen ontvangen wanneer er ongelezen berichten zijn in TPM.** Wanneer er ongelezen berichten zijn in TPM moet de gebruiker hier per e-mail van op de hoogte kunnen worden gebracht. *Motivering:* Door een e-mail te ontvangen hoeft de gebruiker niet regelmatig het systeem te controleren.	–	KD-01 KD-02 KD-04	G
FR-37	**Een gebruiker moet kunnen instellen hoe vaak deze een e-mail van TPM wil ontvangen.** Een gebruiker moet de volgende instelmogelijkheden hebben: a. geen e-mail ontvangen van TPM b. direct een e-mail ontvangen bij een nieuw bericht c. per werkdag een overzicht ontvangen van ongelezen berichten d. per week een overzicht ontvangen van ongelezen berichten. *Motivering:* De mogelijkheid tot het ontvangen van e-mail moet niet als storend worden ervaren, daarom moet de keuze bij de gebruiker liggen.	–	KD-01 KD-02 KD-04	G
FR-38	**TPM moet zijn voorzien van een persoonlijke pagina (dashboard) die door de gebruiker kan worden ingedeeld met behulp van widgets.** Voorbeelden van widgets die aan de het dashboard moeten kunnen worden toegevoegd zijn: • mijn projecten • favoriete projecten • berichten • mijn testomgevingen • favoriete testomgevingen. *Motivering:* Afhankelijk van de rollen die een gebruiker heeft is de informatiebehoefte verschillend. Met behulp van deze persoonlijke pagina kan de gebruiker dit naar wens indelen.	PB-05	KD-01 KD-04	G

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
FR-39	**TPM moet een queryfunctie hebben waarmee binnen de verschillende onderdelen (zoals projecten, testomgevingen en gebruikers) gezocht kan worden.** *Motivering:* Een queryfunctie zorgt ervoor dat de juiste informatie snel te vinden is en maakt geavanceerde zoekopdrachten mogelijk.	PB-05	KD-01 KD-04	G
FR-40	**Er moeten standaard queries beschikbaar zijn waarmee eenvoudige zoekopdracht kunnen worden uitgevoerd.** *Motivering:* Met behulp van queries kunnen ingewikkelde zoekopdrachten worden uitgevoerd. Voor de gebruiksvriendelijkheid is het van belang dat er standaard queries zijn die eenvoudig gebruikt kunnen worden.	PB-05	KD-01 KD-04	G
FR-41	**Een gebruiker moet projecten en testomgevingen als favoriet kunnen markeren.** *Motivering:* Door gebruik te maken van favorieten zijn veel gebruikte onderdelen voor een gebruiker snel benaderbaar.	PB-05	KD-01 KD-04	L
FR-42	**De verschillende functionaliteiten in TPM moeten voorzien zijn van supportpagina's waarin staat beschreven hoe een functie werkt en gebruikt moet worden.** *Motivering:* TPM zal steeds meer functionaliteiten gaan bevatten. Het is belangrijk dat de gebruikers begrijpen hoe deze functionaliteiten werken, zodat ze zelfstandig met de tool om kunnen gaan. Dit zal de inleertijd verlagen.	–	KD-04 US-31	L
FR-43	**Berichten die nieuw zijn sinds de laatste keer dat een gebruiker is ingelogd in TPM moeten gemarkeerd zijn als nieuw.** *Motivering:* Voor de gebruiker moet duidelijk te zien zijn welke berichten er nog bekeken moeten worden.	–	KD-04	L
FR-44	**Datasets moet automatisch geanonimiseerd kunnen worden.** *Motivering:* Vanwege de privacy mag voor het testen niet altijd productiedata gebruikt worden. Door een productiedataset te anonimiseren kan toch productielike data gebruikt worden.	–	US-09	L

Tabel 3.3: Overzicht van de functionele requirements voor het toekomstgerichte testplatform.

3.4.2 Niet-functionele requirements

Een niet-functionele requirement beschrijft een criterium om het functioneren van een systeem te beoordelen. In tabel 3.4 staat een overzicht van de niet-functionele requirements voor het toekomstgerichte testplatform. De requirements zijn opgesplitst in categorieën. De genoemde waarden (zoals percentages en tijden) zijn genoemd ter indicatie. In een vervolgonderzoek (zie paragraaf 7.1) zullen de exacte waarden moeten worden bepaald.

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
	Gebruiksvriendelijkheid			
NFR-01	**Het softwaresysteem moet een herkenbare en makkelijk te onthouden naam krijgen, zodat deze als begrip bekend wordt.** *Motivering:* Een goede naamgeving is belangrijk, omdat dit het centrale systeem zal worden dat gebruikt wordt bij het testen van software. TPM is de werktitel die in dit onderzoek gebruikt is.	–	KD-04	H
NFR-02	**TPM moet voorzien zijn van een Nederlandstalige gebruikersinterface en handleiding.** Alle opgeleverde software moet in ieder geval over een Nederlandstalige handleiding beschikken. *Motivering:* Bij DUO is Nederlands de voertaal.	–	KD-04	H
NFR-03	**TMP moet te gebruiken zijn vanaf mobiele apparaten (zoals mobiele telefoons en tablets).** *Motivering:* Momenteel wordt al veel gebruik gemaakt van mobiele apparaten. In de toekomst zal dit gebruik nog verder toenemen waardoor het belangrijk is dat TPM hier geschikt voor is.	–	KD-04 newline US-14	G
	Beschikbaarheid			
NFR-04	**TPM moet alleen via het intranet te benaderen zijn.** *Motivering:* TPM is geen dienst naar buiten toe en dit verlaagt het risico op aanvallen van hackers.	–	KD-03	H
NFR-05	**TPM moet tussen 7:00 – 20:00 uur een beschikbaarheidspercentage hebben van 99%.** Dit betekent dat de downtime van TPM tussen 7:00 – 20:00 uur een totale duur mag hebben van 15 minuten per dag. *Motivering:* TPM krijgt een belangrijke rol bij DUO. Het niet beschikbaar zijn van TPM zal dan ook invloed hebben op het werk van een grote groep medewerkers.	PB-09	KD-03 US-12 US-18 US-31	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
NFR-06	**De overzetting tussen twee testomgevingen mag niet langer dan 30 minuten duren.** Overzettingen moeten op een efficiënte manier worden uitgevoerd, waarbij er met de planning rekening mee kan worden gehouden wanneer een overzetting uiterlijk gereed is. *Motivering:* Het is van belang dat taken die regelmatig uitgevoerd worden zo efficiënt mogelijk zijn.	PB-09	KD-03 US-08 US-13 US-17 US-24	H
NFR-07	**In de periode van 7:00 – 20:00 uur moet een testomgeving afhankelijk van de categorie één van de volgende beschikbaarheidspercentages hebben:** **a. 90%** **b. 95%** **c. 99%** **Overzettingen worden niet als downtime gerekend.** Dit betekent dat de downtime van een testomgeving tussen 7:00 – 20:00 uur een totale duur mag hebben van: • 144 minuten per dag bij 90% • 72 minuten per dag bij 95% • 15 minuten per dag bij 99%. *Motivering:* Tijdens werkuren moeten de testomgevingen zoveel mogelijk beschikbaar zijn. Een testomgeving die niet beschikbaar is kan invloed hebben op de planning van een één of meerdere projecten. Het beschikbaarheidspercentage is afhankelijk van hoe belangrijk een testomgeving is.	PB-09	KD-03 US-12 US-18 US-24 US-28	H
NFR-08	**In de periode van 20:00 – 7:00 uur hoeft standaard geen minimum beschikbaarheid van een testomgeving te gelden. Het moet wel mogelijk zijn om te kiezen voor eenzelfde beschikbaarheidspercentage als bij NFR-07.** *Motivering:* In de avonden zal normaliter een testomgeving niet gebruikt worden. Dit stelt technisch beheerders in staat om onderhoud te plegen aan het testplatform. Er zijn echter situaties denkbaar waarbij een testomgeving in de avonden wel gebruikt wordt, zoals het testen van een service die een batch uitvoert op deze tijdstippen.	PB-09	KD-03 US-12 US-18 US-24 US-28	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
NFR-09	**Supporttickets moeten binnen één werkdag worden behandeld. Bij een ticket die betrekking heeft op een testomgeving is de reactietijd op werkdagen tussen 7:00 - 20:00 uur afhankelijk van de categorie:** **a. binnen 10 werkuren.** **b. binnen 6 werkuren.** **c. binnen 3 werkuren.** Bij testomgevingen met een hoge beschikbaarheid heeft de support ook een hogere prioriteit. *Motivering:* Projecten lopen risico's wanneer niet tijdig op supporttickets wordt gereageerd.	PB-09	KD-03 US-08 US-24	H
	Interfaces			
NFR-10	**TPM moet voor de meest belangrijke functionaliteiten (zoals het beheer van testomgevingen, overzettingen en relaties) interfaces naar buiten leveren.** De interfaces zouden met behulp van services gerealiseerd kunnen worden. *Motivering:* Het is belangrijk dat andere systemen van de functionaliteiten van TPM gebruik kunnen maken. Zo zou een third-party applicatie afhankelijkheden kunnen herkennen binnen Java-software en deze automatisch kunnen registreren bij TPM. Daarnaast kan deze interface gebruikt worden om functionaliteiten van TPM te gebruiken voor het overzetten van ontwikkeling naar een omgeving binnen het testplatform en vanuit het testplatform naar productie.	VP-03	KD-02	H
NFR-11	**Third-party tools die worden gebruikt (voor taken zoals overzettingen) moeten over een interface beschikken zodat deze door TPM kunnen worden aangestuurd.** *Motivering:* Omdat de gebruikte tools via TPM worden aangestuurd is het belangrijk dat de software over een interface beschikt die door TPM gebruikt kan worden.	VP-03	KD-02	H
	Schaalbaarheid			
NFR-12	**De testomgevingen moeten virtuele omgevingen zijn.** *Motivering:* Door gebruik te maken van virtualisatie kan eenvoudig worden geschaald en zijn omgeving snel aan te maken en weer te verwijderen.	PB-02	KD-02 US-20	H

ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
NFR-13	**Het RAM-geheugen van een testomgeving moet te schalen zijn.** De volgende groottes moeten ondersteund worden: 512 MB, 1024 MB, 2048 MB, 4096 MB, 8192 MB, 16,4 GB of 32,8 GB. *Motivering:* Afhankelijk van de toepassing van de omgeving is een andere hoeveelheid geheugen vereist.	PB-02	KD-02 US-11 US-25	H
NFR-14	**Het aantal CPU-cores van een testomgeving moet te schalen zijn.** De volgende aantallen moeten ondersteund worden: 1, 2, 3, 4, 5 of 6 CPU-cores. *Motivering:* Afhankelijk van de toepassing van de omgeving kan het gewenste aantal CPU-cores verschillen.	PB-02	KD-02 US-11 US-25	H
NFR-15	**De hardeschijfruimte van een testomgeving moet naar beneden of boven te schalen zijn.** De volgende groottes moeten ondersteund worden: 10, 20, 30, 40, 50, 60, 80, 100, 120, 160, 200, 240, 320, 400, 480, 560 of 640 GB. *Motivering:* Afhankelijk van de toepassing van zal de benodigde hoeveelheid hardeschijfruimte verschillen.	PB-02	KD-02 US-11 US-25	H
	Performance			
NFR-16	**De kloksnelheid van één CPU-core voor een testomgeving moet minimaal 2,4 Ghz zijn.** *Motivering:* De meeste processoren die momenteel op de markt zijn voldoen aan deze requirement en dit is belangrijk om te weten voor NFR-14.	–	–	H
	Onderhoudbaarheid			
NFR-17	**Het aantal verschillende soorten voorzieningen waaruit gekozen kan worden bij de aanvraag van een testomgeving (FR-06) moet beperkt blijven.** *Motivering:* Technische beheerders moeten de verschillende soorten testomgevingen goed kunnen beheren en daardoor overzicht houden.	–	US-23	H

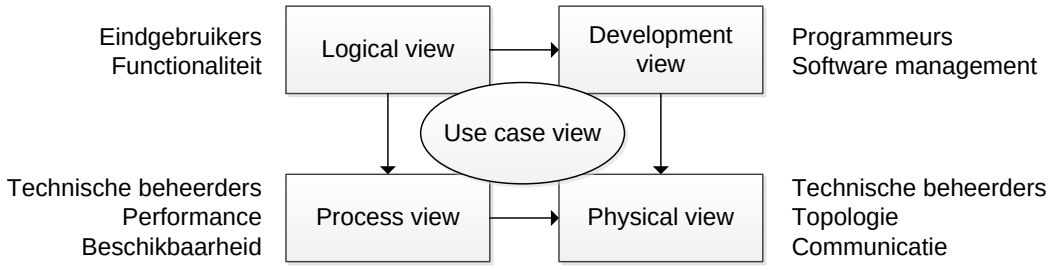
ID	Naam en omschrijving	Probleem- / verbeterpunt	User story / key driver	Prioriteit
NFR-18	**Koppelingen naar de externe softwarepakketten die onderdeel zijn van TPM moeten duidelijk gescheiden zijn van de rest van de software.** *Motivering:* TPM zou gebruik kunnen maken van verschillende third-party tools. Het moet mogelijk zijn deze software eenvoudig te upgraden of te veranderen zonder dat dit grote impact heeft op TPM.	–	–	H
NFR-19	**In TPM moeten alle (Scrum-)projecten die van testomgevingen gebruik maken komen te staan.** *Motivering:* Om van alle mogelijkheden die TPM biedt gebruik te kunnen maken is het belangrijk dat alle projecten van dit centrale systeem gebruik maken.	PB-01	–	G
NFR-20	**TPM moet voldoen aan de webstandaarden voor HTML5 en CSS3.** *Motivering:* Door aan de webstandaarden te voldoen is er minder onderhoud nodig om TPM op de verschillende webbrowsers werkend te krijgen.	–	–	G

Tabel 3.4: Overzicht van de niet-functionele requirements voor het toekomstgerichte testplatform.

4 Architectuur

De huidige en gewenste situatie zijn in de vorige twee hoofdstukken beschreven. Met deze informatie is het mogelijk de bijbehorende architectuur, tooling en beheersaspecten te onderzoeken. In dit hoofdstuk staat de architectuur beschreven die nodig is voor het toekomstgerichte testplatform.

Om de verschillende stakeholders van het project een duidelijk beeld te geven van de architectuur, is gekozen voor het "4+1" view model [10]. Dit model bestaat uit vijf afzonderlijke views: de logical view, process view, physical view, development view en use case view (zie figuur 4.1). De views geven verschillende invalshoeken op de architectuur [11] en kunnen daardoor voor verschillende stakeholders dienen [12]. De komende paragrafen geven een uitleg en uitwerking van elk van de views.

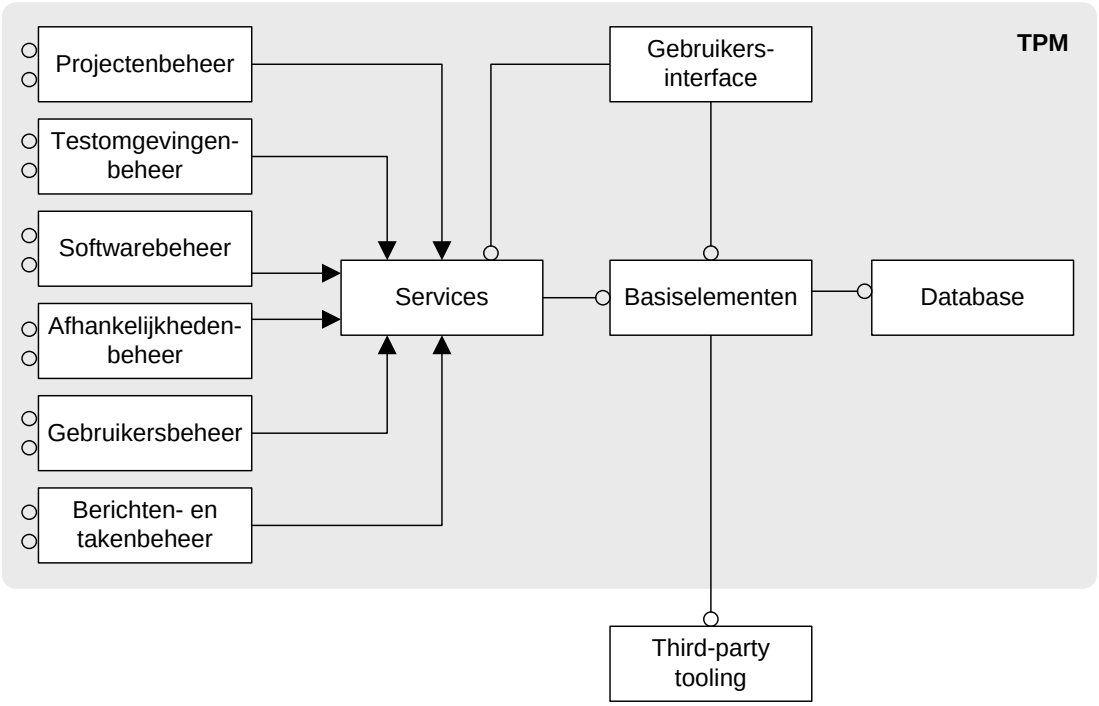


Figuur 4.1: De vijf views van het "4+1" view model met bijbehorende stakeholders.

4.1 Logical view

De logical view ondersteunt voornamelijk de functionele requirements van het testplatform. Met andere woorden: wat het testplatform in de vorm van diensten moet leveren aan haar gebruikers. De view geeft een opsplitsing van de belangrijkste elementen van het testplatform weer in de vorm van objecten of objectklassen. Deze decompositie dient niet alleen als functionele analyse, maar helpt ook bij het identificeren van gemeenschappelijke mechanismen en ontwerpelementen in de verschillende delen van het testplatform.

Figuur 4.2 bevat de logical view van het testplatform waarin de klassengroepen en de gebruikers- en overervingsrelaties staan weergegeven. De objectklassen binnen het grijze vak zijn onderdeel van TPM en daarmee maatwerksoftware.



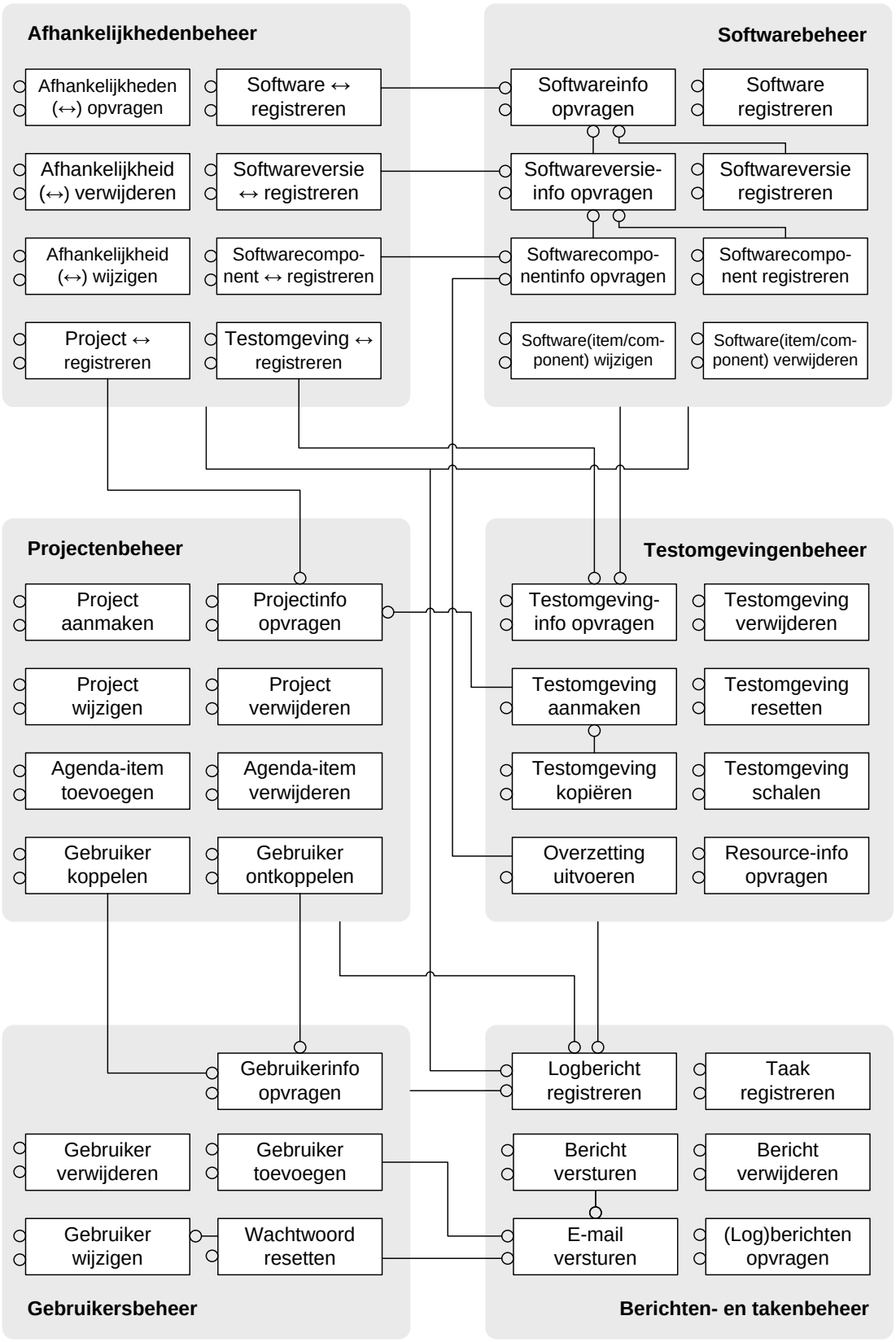
Figuur 4.2: De logical view van het testplatform. Een —∘ geeft een gebruiksrelatie aan en een → geeft een overervingsrelatie aan tussen klassengroepen.

TPM bevat een klassengroep basiselementen die de basisvoorzieningen biedt, zoals communicatie met de database en third-party tooling. Voor het verwerken van gebruikersverzoeken maakt de gebruikersinterface klassengroep gebruik van de basiselementen en services. In de toekomst kan TPM van extra functionaliteit worden voorzien door het toevoegen van nieuwe services. Figuur 4.3 bevat een decompositie van de services klassengroepen. De klassengroep testomgevingenbeheer heeft bijvoorbeeld services voor het:

- opvragen van testomgevingen (al dan niet gefilterd)
- aanmaken, verwijderen, schalen of kopiëren van een testomgeving
- uitvoeren van een overzetting tussen twee testomgevingen.

De services moeten niet alleen binnen het domein van deze architectuur te benaderen zijn. Andere systemen kunnen er ook gebruik van maken (vandaar de losse ∘'s in figuur 4.2 en 4.3). Een voorbeeld daarvan is een oplossing die afhankelijkheden tussen programmatuur herkent en deze automatisch registreert bij een service uit de klassengroep afhankelijkhedenbeheer.

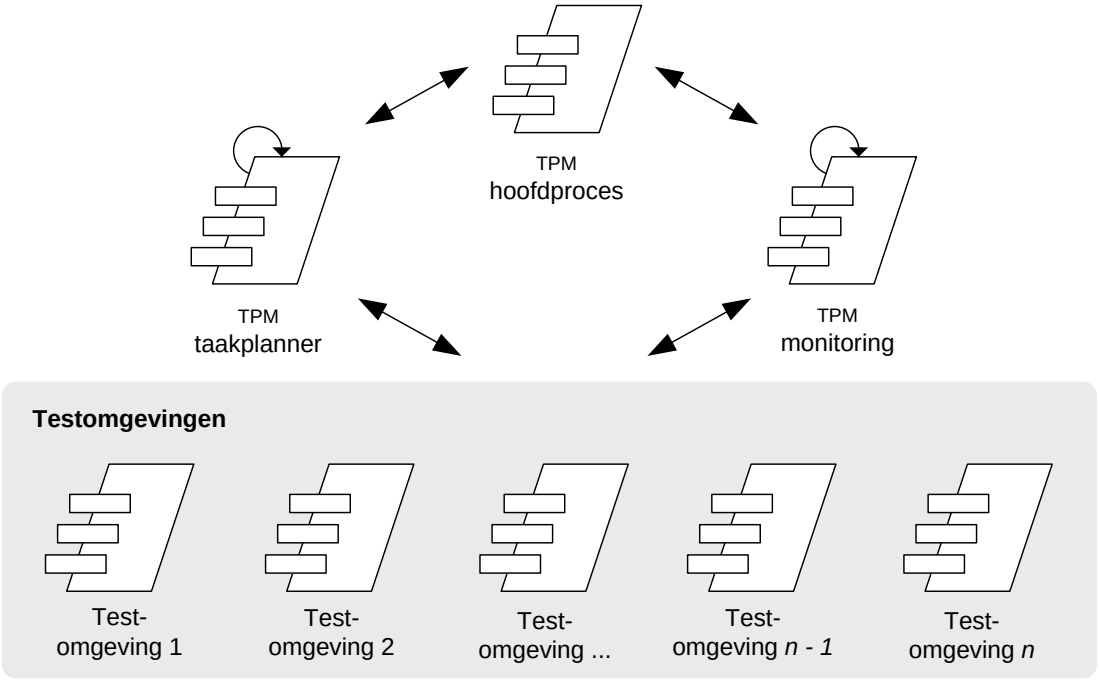
TPM moet systemen buiten de architectuur kunnen aanspreken. Een voorbeeld hiervan is het aanmaken van een testomgeving of het uitvoeren van een overzetting. In plaats van handmatig een third-party tool te gebruiken, kan dit door het testplatform automatisch worden gedaan. Dit verbetert de gebruikerservaring, verhoogt de snelheid van werken en verlaagt de kans op het maken van fouten.



Figuur 4.3: Globale decompositie van de services klassengroepen (grijze vlakken). De decompositie geeft de relaties tussen de services weer.

4.2 Process view

In de process view wordt rekening gehouden met de niet-functionele requirements van het testplatform, zoals de beschikbaarheid en performance. Figuur 4.4 bevat de process view voor het testplatform. De processen, cyclische processen en berichtgevingen tussen de processen staan hierin aangegeven.



Figuur 4.4: De process view van het testplatform. Een ↔ geeft bidirectionele berichtgeving aan en een ↻ geeft een cyclisch proces aan.

In het hoofdproces draait de TPM webapplicatie. Dit proces wordt niet door TPM zelf aangemaakt, maar valt onder het beheer van de webserver. Sommige webservers maken voor elke verbinding een apart proces aan (zoals Apache en Microsoft IIS), terwijl anderen slechts enkele processen gebruiken om toch veel gebruikersverzoeken af te handelen (zoals Nginx).

De taakplanner is een cyclisch, multithreaded proces dat regelmatig controleert of er taken in de wachtrij staan voor het testplatform. Elke taak dient in een eigen thread uitgevoerd te worden. Voorbeelden van taken zijn:

- het aanmaken of verwijderen van een testomgeving
- het doorvoeren van een wijziging op een testomgeving
- het uitvoeren van een overzetting tussen twee testomgevingen.

Er is voor een apart proces gekozen om de beschikbaarheid en performance van het hoofdproces op een hoog niveau te houden. Als de taken door het hoofdproces uitgevoerd worden, dan zou dit andere gebruikers van TPM kunnen hinderen.

Het monitoringproces houdt periodiek de resources van testomgevingen in de gaten, zoals het CPU-, geheugen- en hardeschijfgebruik. Ook stuurt het proces regelmatig heartbeatmessages uit om te controleren of testomgevingen nog juist werken. Meldingen worden gerapporteerd in TPM. Het proces is net als de taakplanner cyclisch en multithreaded uitgevoerd.

Abstract gezien draait elke testomgeving als een eigen proces. Feitelijk draait elke testomgeving op een eigen besturingssysteem dat is opgebouwd uit tientallen processen. In tegenstelling tot de TPM-processen (hoofdproces, taakplanner en monitoring), zal het voorkomen dat een testomgeving (en dus een abstract proces) wordt aangemaakt, gestart, gestopt, herstart of verwijderd.

4.3 Physical view

De physical view geeft de architectuur weer vanuit het oogpunt van een technisch beheerder. Het behandelt de topologie van het systeem en niet-functionele requirements als schaalbaarheid en performance.

In de probleemstelling (paragraaf 1.2) is een definitie van een testomgeving gegeven, namelijk: “een combinatie van hardware en software waarop een test kan worden uitgevoerd”. De requirements bepalen deels welke architectuur nodig is voor het testplatform. Deze geven aan dat testomgevingen binnen een afgesproken tijd geleverd moeten worden, uit bepaalde voorzieningen gekozen moet kunnen worden en dat omgevingen eenvoudig schaalbaar moeten zijn. Verder staat er in de concept ICT-principes (bijlage E) dat er gevirtualiseerd moet worden. Deze punten maken cloud computing een logische keuze.

Cloud computing

De term cloud computing kent verschillende definities in literatuur en op het internet. Voor het beschrijven van deze term is bij dit onderzoek uitgegaan van de definitie die door de NIST (National Institute of Standards and Technology) is opgesteld [13]. De belangrijkste kenmerken van cloud computing zijn:

- **On-demand zelfservice:** Een gebruiker kan geautomatiseerd computerfaciliteiten (zoals rekencapaciteit en opslagruimte) aanvragen en tot zijn beschikking krijgen.
- **Bereikbaar via netwerk:** De computerfaciliteiten zijn via verschillende apparaten (thick- en thin-clients) via een netwerk toegankelijk, waarbij gebruik wordt gemaakt van standaardtechnieken.
- **Locatieonafhankelijk:** De resources worden dynamisch toegewezen en kunnen zich op verschillende locaties bevinden. Er kan een mogelijkheid worden geboden waarbij de afnemer het gewenste land of datacenter kan kiezen.
- **Eenvoudig schaalbaar:** De faciliteiten kunnen eenvoudig en op elk moment worden geschaald. In sommige gevallen kan de capaciteit automatisch wor-

den vergroot of verkleind op de momenten dat dit nodig is.
- **Meetbare services:** De geleverde diensten (zoals opslag-, bandbreedte- en CPU-gebruik) worden gemonitord. Deze informatie is voor zowel de aanbieder als de afnemer inzichtelijk. Hierdoor kan worden bepaald wat voor een dienst moet worden betaald.

De NIST definieert drie servicelagen waarop clouddiensten kunnen worden aangeboden:

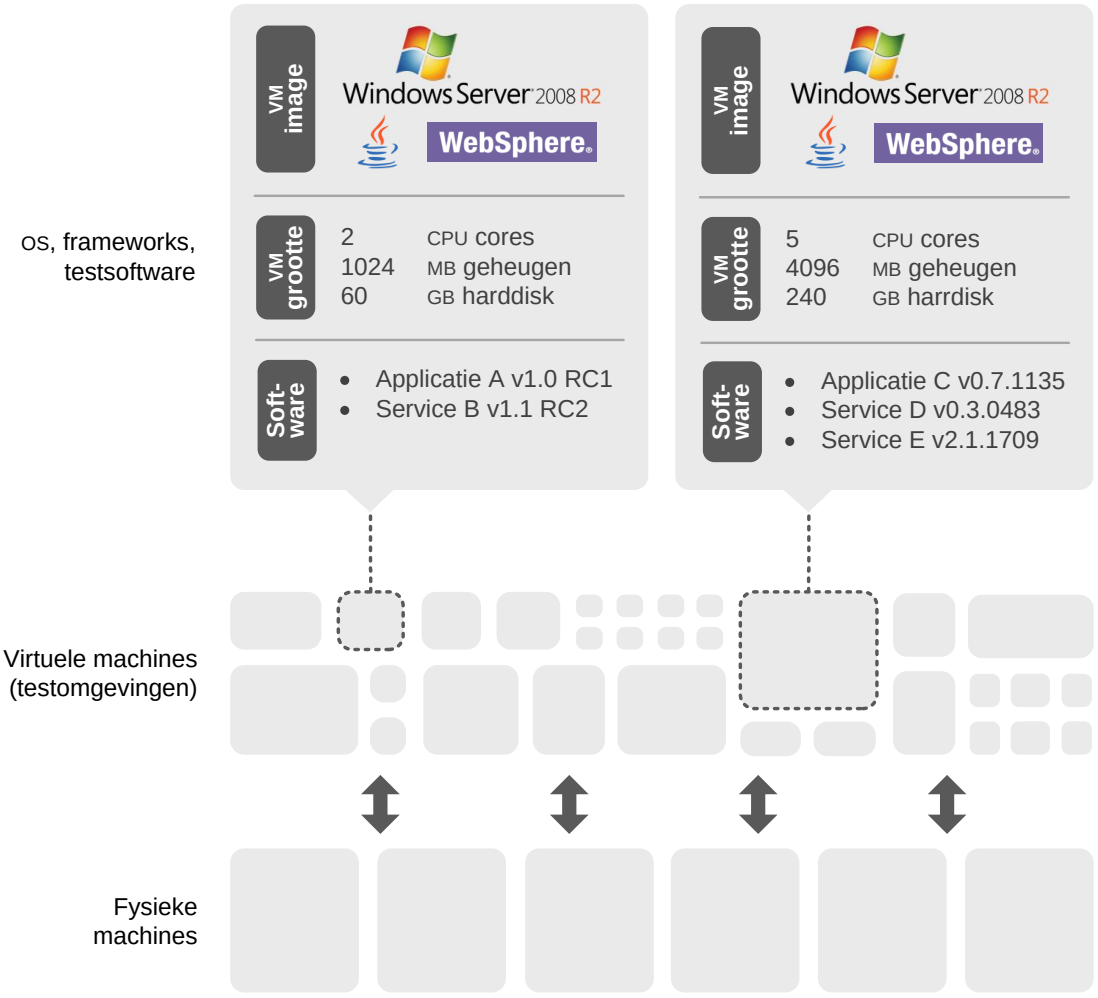
- **IaaS (Infrastructure as a Service)**: Deze laag geeft de afnemer de mogelijkheid om het besturingssysteem, de applicaties en de opslagruimte zelf te beheren. De overige infrastructuurcomponenten, zoals het netwerk en de servers, worden beheerd door de aanbieder. Het is in sommige gevallen wel mogelijk om netwerkcomponenten (zoals een firewall) zelf te configureren.
- **PaaS (Platform as a Service)**: Deze laag geeft de mogelijkheid om eigen applicaties op een cloudinfrastructuur te draaien. Hierbij kan de afnemer de applicatieomgeving zelf configureren. Het beheer van de onderliggende netwerken, servers, besturingssystemen en opslagruimte is echter de verantwoordelijkheid van de aanbieder.
- **SaaS (Software as a Service)**: Is het aanbieden van applicaties op een cloudinfrastructuur. De gebruiker kan de software gebruiken via bijvoorbeeld een webinterface. De afnemer hoeft zich geen zorgen te maken over het beheer en de configuratie van onderliggende netwerken, servers, besturingssystemen en opslagruimte.

Voor de infrastructuur en inrichting van de cloud kan uit verschillende soorten oplossingen worden gekozen. Deze oplossingen hebben de volgende kenmerken:

- **Private cloud:** Een cloudinfrastructuur voor exclusief gebruik door een enkele organisatie. Het beheer kan worden gedaan door de organisatie zelf, een externe partij of een combinatie van beiden. De hosting kan in een eigen datacenter of door een externe partij worden gedaan.
- **Community cloud:** Een cloudinfrastructuur voor exclusief gebruik door een specifieke groep van organisaties met gedeelde belangen (zoals beveiligingseisen). De cloud is eigendom van en wordt beheerd door één of meer van de deelnemende organisaties, een externe partij, of een combinatie van beiden. De hosting kan zowel intern als extern worden gedaan.
- **Public cloud:** Een cloudinfrastructuur voor publiek gebruik. De cloud is eigendom van en wordt beheerd door een commerciële-, academische- of overheidsorganisatie, of een combinatie daarvan. De cloud wordt gehost in het datacenter van de aanbieder.
- **Hybrid cloud:** Een cloudinfrastructuur die bestaat uit een samenstelling van twee of meer van bovenstaande cloudinfrastructuren. Deze clouds behouden hun unieke kenmerken, maar zijn middels eigen of gestandaardiseerde technieken met elkaar verbonden.

Testomgevingen in de cloud

Bij DUO wil een afnemer van een testomgeving binnen deze omgeving zijn eigen applicaties kunnen testen. Hierdoor is het belangrijk dat er kan worden gekozen voor een applicatieserver die geschikt is voor een bepaalde versie van Java en dat deze de juiste frameworks bevat. Het beheer van de onderliggende lagen mag door een andere partij worden gedaan, waardoor PaaS een geschikte oplossing is. In figuur 4.5 is de physical view van het testplatform weergegeven.



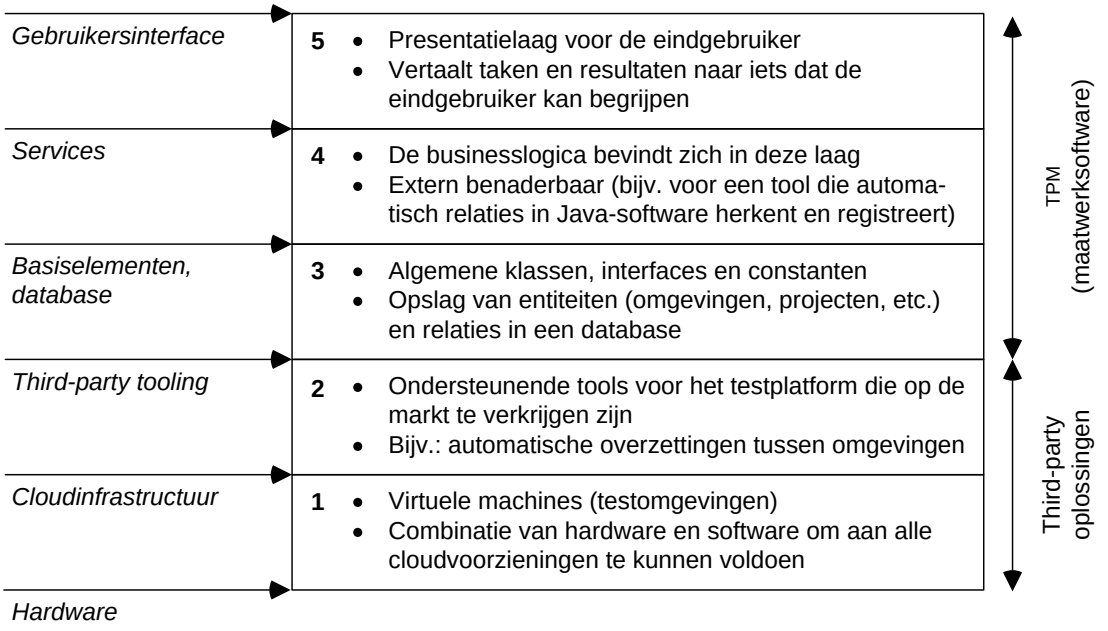
Figuur 4.5: De physical view van het testplatform. Testomgevingen worden als PaaS aangeboden, wat het beheren en schalen eenvoudiger maakt.

Naast de servicelaag moet ook het soort cloud worden bepaald. De cloud is op dit moment nodig voor een enkele organisatie (alleen voor DUO), waarbij het cloudbeheer door de organisatie zelf kan worden gedaan. De concept ICT-principes geven aan dat nieuwe systemen moeten worden aangeboden vanuit het eigen datacenter in Groningen. Deze punten zorgen ervoor dat een private cloud de meest geschikte keuze is.

4.4 Development view

De development view concentreert zich op de opbouw van de software. De software is doorgaans opgesplitst in kleinere delen, ook wel subsystemen genoemd. Deze subsystemen kunnen door verschillende ontwikkelaars gemaakt worden en zijn vaak in lagen in te delen. Een laag kan één of meerdere subsystemen bevatten en levert aan de bovenliggende lagen een duidelijke en afgebakende interface. Een subsysteem mag alleen afhankelijk zijn van subsystemen in diezelfde of onderliggende lagen.

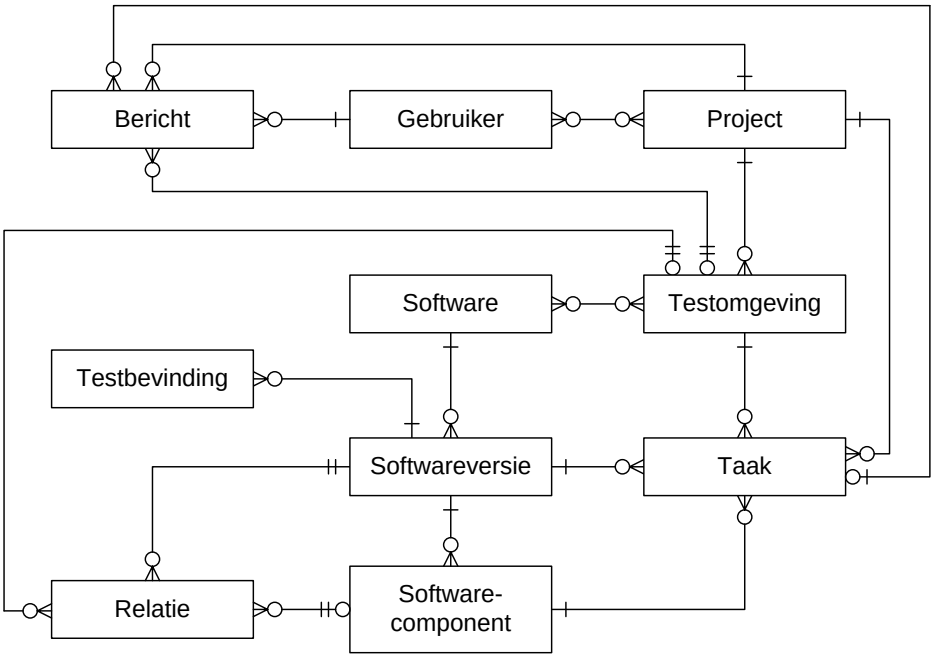
Figuur 4.6 geeft de development view van het testplatform weer, ingedeeld in vijf lagen. De eerste twee lagen zijn third-party oplossingen die niet zelf ontwikkeld hoeven te worden (zie paragraaf 5.1). De bovenste drie lagen vormen samen TPM, dat wél maatwerksoftware is (zie paragraaf 5.2). De basiselementen (derde laag) bestaan uit algemene klassen, interfaces en constanten die door de services en gebruikersinterface kunnen worden gebruikt. De services (vierde laag) bevatten de businesslogica en leveren daarmee de meeste functionaliteiten voor het testplatform. De services zijn voor systemen buiten TPM benaderbaar. De gebruikersinterface (vijfde laag) dient als presentatielaag naar de eindgebruiker.



Figuur 4.6: De development view van het testplatform. Elke laag bevat één of meerdere subsystemen. De bovenste drie lagen vormen samen TPM.

De database (derde laag) is een afzonderlijk subsysteem waarin de entiteiten en relaties van het testplatform wordt opgeslagen. Figuur 4.7 geeft het ERD (Entity-Relationship Diagram) van de database weer. Het bevat entiteiten zoals projecten, testomgevingen, gebruikers, software en softwarecomponenten (bijvoorbeeld klassen, functies en bestanden). Het ERD bevat ook de relaties tussen deze en-

titeiten met een bijbehorende kardinaliteit. Een voorbeeld daarvan is de relatie tussen een project en een testomgeving. Een project kan nul of meerdere testomgevingen bevatten en een testomgeving hoort altijd bij één project. Een ander voorbeeld is de relatie tussen een bericht en een testomgeving. Een bericht kan bij nul, één of twee testomgevingen horen en een testomgeving kan aan nul, één of meerdere berichten gekoppeld zijn.



Figuur 4.7: Het ERD van TPM, waarin de entiteiten, relaties en kardinaliteit voor de database staan vermeld.

4.5 Use case view

In de use case view zijn alle elementen van het "4+1" view model samengevoegd. Deze view beschrijft hoe het systeem als geheel zal werken, waarbij een aantal belangrijke scenario's met behulp van use cases zijn uitgewerkt. Op deze manier wordt voor de stakeholders duidelijk hoe in de nieuwe situatie met het systeem zal worden gewerkt. In een aantal use cases wordt gesproken over de beherende afdeling van het testplatform. Deze afdeling is verder toegelicht in hoofdstuk 6.

Aanvragen van een testomgeving

Bij een project wordt gebruik gemaakt van testomgevingen, waarbij het aantal en het soort testomgevingen per project kan verschillen. De benodigde testomgevingen moeten daarom gemakkelijk via TPM kunnen worden aangevraagd. De stappen die hierbij worden doorlopen staan in use case 4.1 beschreven.

	Aanvragen van een testomgeving
Omschrijving	Het aanvragen van een testomgeving voor een project.
Actor	Projectleider.
Preconditie	Het project waar de testomgeving onderdeel van moet worden is reeds in TPM beschikbaar en de projectleider is bepaald.
Basisverloop	1. De projectleider start de aanvraag voor een testomgeving in TPM. 2. De projectleider voert een naam in voor de testomgeving. 3. De projectleider kiest het project waaraan de testomgeving moet worden toegevoegd (een projectleider kan immers verantwoordelijk zijn voor meerdere projecten). 4. De projectleider kiest de cloudimage die voor de testomgeving gebruikt moet worden. 5. TPM toont de kosten van de testomgeving. De projectleider kiest het gewenste aantal CPU's, de hoeveelheid geheugen, de hoeveelheid opslagruimte en de gewenste categorie. Afhankelijk van de keuzes zullen de kosten veranderen. 6. De projectleider bevestigt de aanvraag en de beheerafdeling wordt via TPM op de hoogte gebracht. 7. De beheerafdeling geeft een akkoord via TPM en de testomgeving wordt geautomatiseerd in de cloud aangemaakt. 8. De projectleden worden in TPM op de hoogte gebracht dat de testomgeving gereed is.
Alternatief verloop 1	*De aanvraag wordt afgekeurd:* 7a. De beheerafdeling gaat niet akkoord met de aanvraag en geeft de reden hiervoor aan in TPM. 8a. De projectleider ontvangt een bericht in TPM dat de aanvraag is afgekeurd en wat de reden hiervoor is.
Postconditie	De testomgeving is aangemaakt.
Alternatieve postconditie	De testomgeving is niet geaccepteerd en daardoor niet aangemaakt.

Use case 4.1: Het aanvragen van een nieuwe testomgeving voor een project in TPM en de uitvoering van deze aanvraag.

Overzetting tussen twee testomgevingen

Doordat er met verschillende testomgevingen gewerkt wordt, moet het mogelijk zijn om eenvoudig software tussen testomgevingen over te zetten. Dit moet zoveel mogelijk geautomatiseerd gebeuren. De stappen die uitgevoerd worden bij het aanvragen en uitvoeren van een overzetting tussen twee testomgevingen staan beschreven in use case 4.2.

	Overzetten testomgeving X → testomgeving Y
Omschrijving	Het overzetten van software tussen twee testomgevingen.
Actor	Testcoördinator van testomgeving X.
Precondities	• De software bevindt zich in testomgeving X. • Alle testers hebben aangegeven dat de tests gereed zijn.
Basisverloop	1. De testcoördinator start in TPM de aanvraag voor een overzetting van de software in testomgeving X. 2. De testcoördinator kiest voor een overzetting naar testomgeving Y en geeft de reden voor de overzetting aan. 3. TPM stuurt een bericht naar de projecten die een afhankelijkheid met de gekozen testomgeving hebben. 4. De partijen (zoals de tester van testomgeving Y) die een overzetting moeten accepteren worden geïnformeerd.*In* TPM *is altijd inzichtelijk welke partijen nog een beoordeling moeten geven.* 5. De betreffende partijen geven in TPM een akkoord voor de overzetting. 6. De overzetaanvraag komt in de wachtrij te staan bij de beheerafdeling die de overzettingen uitvoert. 7. Een beheerder start het automatische overzetproces en controleert het resultaat. 8. De betrokken partijen worden via TPM automatisch van de overzetting op de hoogte gebracht.
Alternatief verloop 1	*De overzetting wordt afgekeurd:* 5a. De betreffende partij geeft in TPM geen akkoord voor de overzetting en motiveert deze beslissing. 6a. De testcoördinator van testomgeving X ontvangt een bericht dat de overzetting is afgekeurd en wat de reden hiervoor is.
Alternatief verloop 2	*Het project beschikt over een eigen beheerder:* 6b. De beheerder behorende bij het project voert de automatische overzetting uit en controleert het resultaat. 7b. De betrokken partijen worden via TPM automatisch van de overzetting op de hoogte gebracht.
Postconditie	De overzetting is uitgevoerd en de betrokken partijen zijn hierover geïnformeerd.
Alternatieve postconditie	De overzetting is niet uitgevoerd en de testcoördinator heeft een bericht met de reden ontvangen.

Use case 4.2: Het aanvragen van een overzetting tussen twee testomgevingen in TPM en de uitvoering van deze aanvraag.

Ondersteuning vragen

Een ander belangrijk onderdeel is het vragen van ondersteuning. Afhankelijk van de vraag, het probleem en het type project zijn er verschillende manieren om ondersteuning te vragen. Deze mogelijkheden zijn beschreven in use case 4.3.

	Het vragen van ondersteuning
Omschrijving	Het vragen van ondersteuning.
Actor	Een TPM gebruiker.
Preconditie	De gebruiker heeft een account in TPM.
Basisverloop	1. De gebruiker maakt in TPM een supportticket aan. 2. Een medewerker van de beheerafdeling geeft antwoord op de supportticket. 3. De status van de supportticket wordt veranderd in gesloten.
Alternatief verloop 1	*Telefonische ondersteuning:* 1a. De gebruiker neemt telefonisch contact op via het algemene support telefoonnummer van de beheerafdeling. 2a. Een medewerker van de beheerafdeling beantwoordt de vraag en registreert dit in TPM. 3a. De gebruiker ziet de supportticket betreffende de telefonische ondersteuning terug in TPM met als status gesloten.
Alternatief verloop 2	*De gebruiker heeft een vraag met betrekking tot een project met een eigen beheerder:* 1b. De gebruiker neemt (direct, per telefoon, via TPM) contact op met de bij het project behorende beheerder. 2b. De beheerder lost het probleem op of beantwoordt de vraag. 3b. De status van de supportticket wordt veranderd in gesloten.
Alternatief verloop 3	*De gebruiker heeft een vraag met betrekking tot een project:* 1c. De gebruiker maakt in TPM een supportticket aan en koppelt deze ticket aan het betreffende project. De supportticket is daardoor voor alle projectleden inzichtelijk. 2c. Een medewerker van de beheerafdeling beantwoordt de supportticket en zowel de gebruiker die de ticket heeft aangevraagd als de projectleden worden via TPM van het antwoord op de hoogte gebracht. 3c. De status van de supportticket wordt veranderd in gesloten.
Postconditie	Het probleem is opgelost of de vraag is beantwoord. De supportticket is gesloten.

Use case 4.3: Het verloop van een supportaanvraag met daarbij de verschillende alternatieven.

Verwijderen van een testomgeving

Als een project is afgerond of als een testomgeving slechts tijdelijk nodig was kan een projectleider via TPM aangeven dat de testomgeving moet worden verwijderd. De stappen die hierbij worden doorlopen staan in use case 4.4 beschreven.

	Verwijderen van een testomgeving
Omschrijving	Het verwijderen van een testomgeving die voor een project niet langer nodig is.
Actor	Projectleider.
Preconditie	Er is minimaal één testomgeving beschikbaar binnen de projecten die de projectleider onder zijn beheer heeft.
Basisverloop	1. De projectleider kiest de testomgeving en geeft hierbij aan dat deze verwijderd moet worden. 2. TPM geeft aan of deze testomgeving afhankelijkheden heeft en vraagt om een bevestiging. 3. De projectleider bevestigt dat de testomgeving verwijderd moet worden. 4. De testomgeving krijg de status “Wordt verwijderd” en de beheerafdeling wordt via TPM op de hoogte gebracht. Afhankelijke projecten worden via een bericht in TPM op de hoogte gebracht. 5. De beheerafdeling geeft een akkoord voor het verwijderen en het geautomatiseerde verwijderproces wordt gestart. 6. De projectleden worden in TPM op de hoogte gebracht dat de testomgeving is verwijderd.
Alternatief verloop 1	*De projectleider geeft geen bevestiging:* 3a. De projectleider geeft aan niet door te gaan met het verwijderproces. 4a. Het verwijderproces wordt afgebroken en de testomgeving zal niet worden verwijderd.
Alternatief verloop 2	*De aanvraag wordt afgekeurd:* 5b. De beheerafdeling gaat niet akkoord met het verwijderen en geeft de reden hiervoor aan in TPM. 6b. De projectleider ontvangt een bericht in TPM dat de testomgeving niet verwijderd wordt en wat de reden hiervoor is. De status van de testomgeving zal terug gaan naar zijn voorgaande status en afhankelijke projecten krijgen een bericht dat het verwijderen niet doorgaat.
Postconditie	De testomgeving is verwijderd. De resources zijn vrijgegeven en er worden niet langer kosten in rekening gebracht.
Alternatieve postconditie 1	De testomgeving is niet verwijderd.
Alternatieve postconditie 2	De testomgeving is niet verwijderd en de projectleider heeft een bericht met de reden ontvangen.

Use case 4.4: Het verwijderen van een testomgeving in TPM.

5 Tooling

In het vorige hoofdstuk is met het ontwerp van de architectuur een basis gelegd voor het testplatform. Dit maakt het testplatform echter nog niet compleet, want een groot deel van de requirements vereisen softwarematige oplossingen. Deze oplossingen kunnen worden gerealiseerd met third-party tooling (software van andere leveranciers) en met custom tooling (maatwerksoftware).

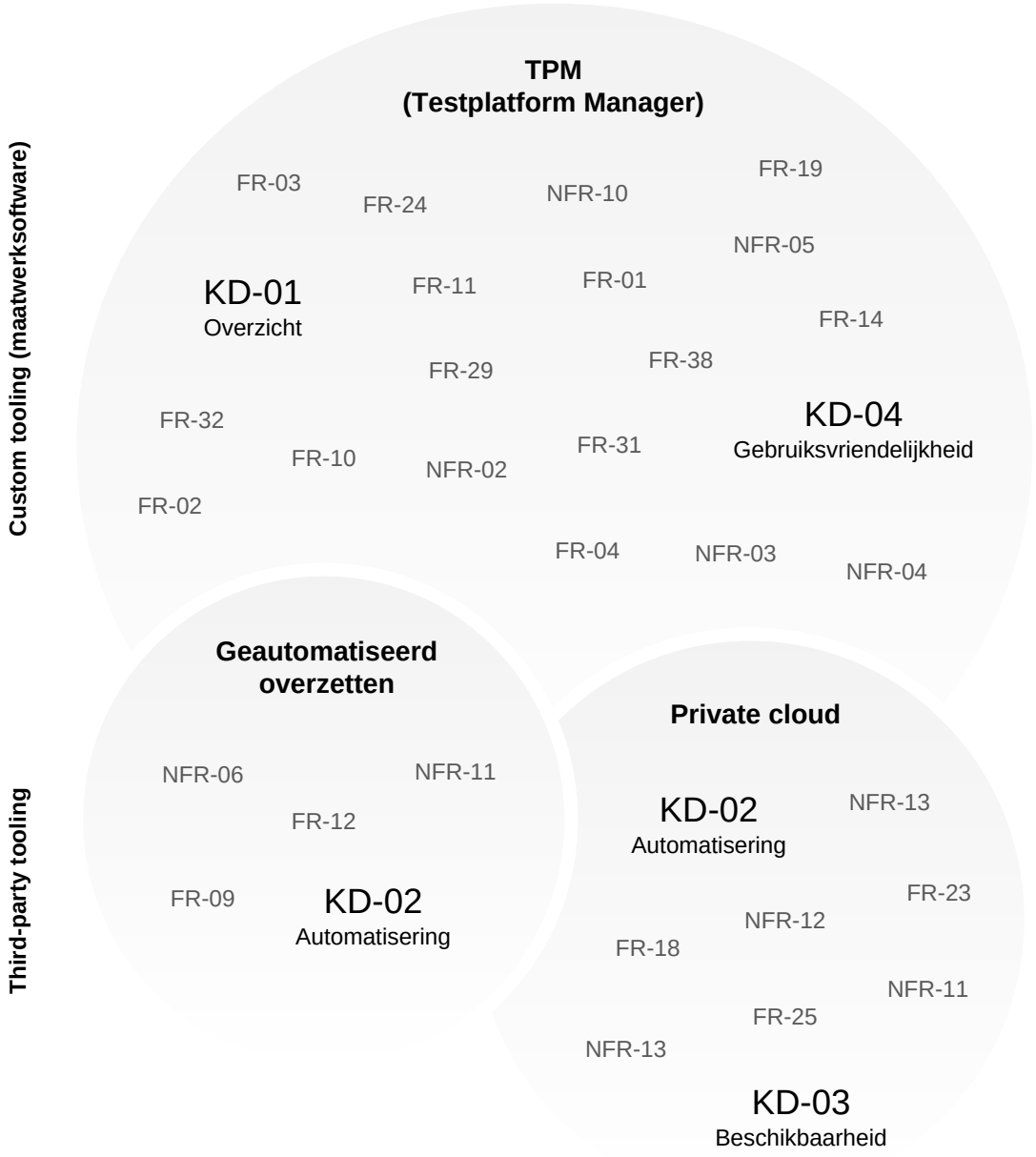
Voor het testplatform is een combinatie van third-party en custom tooling de meest voor de hand liggende oplossing. Het komt zelden voor dat een probleem van dit formaat met alleen third-party tooling kan worden opgelost. Er is ook maatwerksoftware nodig om aan alle wensen van de gebruikers te kunnen voldoen. In de architectuur is rekening gehouden met het ondersteunen van software van andere leveranciers. Dit hoofdstuk beschrijft de ontwerpkeuzes voor zowel de third-party als custom tooling.

5.1 Third-party tooling

Third-party tooling kan in bepaalde gevallen een goed alternatief zijn voor maatwerksoftware. Deze vorm van tooling is door een andere leverancier gemaakt en deze levert hier doorgaans ondersteuning bij. Hoewel er meestal licentie- en ondersteuningskosten moeten worden betaald, worden de hoge ontwikkelkosten van maatwerksoftware bespaard. Een nadeel van third-party tooling is dat deze, in tegenstelling tot maatwerksoftware, vaak moet worden aangepast met speciale configuraties, plugins of zelfs wijzigingen in de broncode door de leverancier.

Vanwege de breedte van dit onderzoek is het niet mogelijk om alle tools die nodig zijn volledig uit te werken. Een aantal tools zullen in een vervolgonderzoek (zie paragraaf 7.1) onderzocht moeten worden. In deze paragraaf zijn de third-party tools voor het opzetten van een private cloud en het uitvoeren van geautomatiseerde overzettingen tussen testomgevingen uitgewerkt. Deze tools dekken een groot deel van de functionele en niet-functionele requirements en op de markt zijn hier oplossingen voor beschikbaar. Van beide type tools zijn mogelijke alternatieven beschreven en er is uitgelegd waarom een bepaalde keuze is gemaakt.

Figuur 5.1 geeft een schematisch overzicht van de tools die zijn onderzocht en welke requirements en key drivers deze moeten dekken. In de figuur is te zien dat de eindgebruiker weinig van de third-party tooling zal merken, omdat TPM er als schil omheen ligt.



Figuur 5.1: De combinatie van third-party tooling en custom tooling (maatwerksoftware). TPM ligt als schil om de third-party tooling heen en is daarmee de enige zichtbare tool voor de eindgebruiker.

5.1.1 Private cloud

In de architectuur is bepaald dat er gebruik zal worden gemaakt van een private cloud voor het virtualiseren van Java applicatieservers. Hiervoor zijn verschillende oplossingen beschikbaar. In deze paragraaf zijn een aantal mogelijkheden beschreven. Hierbij wordt uitgegaan van het gebruik van de IBM WebSphere applicatieserver. Deze keuze is gemaakt omdat DUO kennis en ervaring heeft met deze applicatieserver. Bij het veranderen van deze software zal nieuwe kennis

moeten worden opgedaan, waardoor het toekomstgerichte testplatform onnodig complexer wordt gemaakt.

IBM Workload Deployer

Workload Deployer is een oplossing van IBM om snel virtuele systemen op te zetten in een private cloud [14]. Er kan zowel worden gewerkt met basisimages, die op verschillende manieren kunnen worden aangepast, als met geheel zelf gemaakte images. Van verschillende IBM producten, waaronder de WebSphere applicatieserver, zijn deze basisimages beschikbaar als Hypervisor Editions. Voor de private cloud worden verschillende hypervisors ondersteund, waaronder VMWare ESX. DUO heeft ervaring met virtualisatie met behulp van VMWare.

Definitie van Hypervisor

Een hypervisor, ook wel virtual machine manager genoemd, is een hardware virtualisatietechniek die het mogelijk maakt om meerdere besturingssystemen tegelijkertijd op een host computer te laten draaien.

Workload Deployer bestaat uit een hardwaremodule die gebaseerd is op de IBM DataPower 7199 / 9005. De hardware met bijbehorende software heeft verschillende mogelijkheden om een hoge beschikbaarheid te garanderen en kan worden geplaatst in een eigen datacenter. Via een Web 2.0 gebruikersinterface, een CLI (Commandline Interface) en REST API's is Workload Deployer aan te sturen. Integratie met andere systemen is daardoor mogelijk.

Naast de mogelijkheid van het aanpassen van images beschikt Workload Deployer over zogenaamde patterns en scripts. Met behulp daarvan kunnen virtuele systemen automatisch worden ingesteld, zodat deze snel kunnen worden aangemaakt en de standaardconfiguratie kan worden toegepast. Ook het verwijderen, kopiëren, monitoren van resources en het maken van backups (snapshots) van een virtueel systeem behoort tot de mogelijkheden van Workload Deployer.

Bovenstaande functionaliteiten dekken een groot deel van de requirements. Hierdoor is IBM Workload Deployer in combinatie met VMWare ESX een geschikte keuze voor het opzetten van een private cloud in een eigen datacenter.

Amazon VPC

Amazon VPC (Virtual Private Cloud) [15] biedt de mogelijkheid om een virtuele private cloud op te zetten op de servers van Amazon. Zoals de naam al aangeeft gaat het om een virtuele private cloud, waardoor deze oplossing niet volledig voldoet aan de definitie van een private cloud (zie paragraaf 4.3). Bij een VPC is de cloud virtueel opgedeeld, waardoor de fysieke cloudinfrastructuur niet door één exclusieve organisatie wordt gebruikt.

Binnen deze cloud kan gebruik worden gemaakt van de Amazon EC2 (Elastic Compute) [16] web services, waardoor met behulp van een AMI (Amazon Machine Image) eenvoudig servers kunnen worden aangemaakt. De IBM WebSphere applicatieserver is beschikbaar als AMI, maar het is ook mogelijk eigen images te maken. Voor de integratie met andere systemen zijn API's beschikbaar.

Met Amazon kan er snel van een private cloud gebruik worden gemaakt zonder dat hiervoor direct grote investeringen nodig zijn. De virtuele servers hebben monitoring, zijn schaalbaar en er moet worden betaald voor datgene wat wordt gebruikt.

IBM SmartCloud Application Services

IBM SmartCloud Application Services [17] is een PaaS oplossing specifiek bedoeld voor bedrijven. Hierbij wordt gebruik gemaakt van IBM SmartCloud Enterprise. De cloud heeft overeenkomsten met zowel Workload Deployer als de Amazon cloud. De IBM cloud biedt namelijk grotendeels dezelfde functionaliteiten als Workload Deployer, alleen is er net zoals bij Amazon geen eigen datacenter nodig (de cloud draait in de datacenters van IBM). De SmartCloud is een VPC en images die geschikt zijn voor Workload Deployer kunnen hier eenvoudig naar worden gedeployed.

SmartCloud Application Services heeft veel overeenkomsten met Amazon, maar heeft één belangrijk voordeel. Er wordt gewerkt op een manier die compatibel is met Workload Deployer. Dit biedt de mogelijkheid om eerst te kiezen voor de VPC oplossing, waarbij gebruik wordt gemaakt van de SmartCloud. Zo kan zonder grote investering met een cloud gewerkt worden. In de toekomst kan dan worden besloten om met behulp van Workload Deployer een cloud in een eigen datacenter op te zetten.

Ontwerpkeuze

Voor DUO is IBM Workload Deployer de beste oplossing. De reden hiervoor is dat DUO over een eigen datacenter en beheerafdeling beschikt, er al ervaring is met virtualisatie (VMware) en dat de concept ICT-principes voorschrijven dat systemen vanuit het eigen datacenter in Groningen aangeboden moeten worden. Er wordt dan gebruik gemaakt van een echte private cloud en geen VPC oplossing. Op deze manier heeft DUO alles in eigen beheer.

Om niet direct een grote investering te hoeven doen en eerst kennis over het werken met een cloud op te doen, kan IBM SmartCloud Application Services tijdelijk gebruikt worden. De compatibiliteit met Workload Deployer is een pré. Amazon VPC is voor DUO het minst geschikt.

5.1.2 Geautomatiseerd overzetten

Een belangrijk onderdeel van het testplatform is het geautomatiseerd overzetten tussen testomgevingen, ook wel software deployment genoemd. Het gaat hierbij om alle stappen die op de juiste volgorde uitgevoerd moeten worden om de software werkend te krijgen, zodat de eindgebruiker deze in gebruik kan nemen. Naast het overzetten van de software zelf houdt dit ook in dat de juiste data beschikbaar is en instellingen juist zijn geconfigureerd.

Met het automatiseren van het software deploymentproces kan veel tijd worden bespaard. Ook kunnen fouten veroorzaakt door menselijke handelingen worden voorkomen. Uit onderzoek is gebleken dat een Java-ontwikkelaar die WebSphere gebruikt gemiddeld circa 250 uur (15% van de werkuren) per jaar bezig is met het deploymentproces [18]. Dit toont aan dat hier veel tijdswinst te behalen valt.

Deployit

Bij DUO loopt sinds mei 2011 een project voor het automatisch overzetten. Voor dit project is Deployit van XebiaLabs [19] aangeschaft. Deze software heeft als doel het automatiseren van het volledige deploymentproces om daarmee tijd en kosten te besparen. Er is ondersteuning voor een groot aantal applicatieservers, databases en berichtengines. Hierdoor is het mogelijk om in de toekomst bijvoorbeeld een ander type database te gebruiken, zonder dat dit problemen oplevert voor de automatische deployment. Er wordt steeds meer ondersteuning toegevoegd, zo is er naast Java ook ondersteuning voor Microsoft .NET. Verschillende grote organisaties, waaronder KLM, maken gebruik van deze software.

Deployit beschikt naast een webinterface over een CLI, waardoor integratie in andere systemen mogelijk is. Om ervoor te zorgen dat Deployit binnen verschillende organisaties toepasbaar is, wordt gewerkt met een pluginsysteem. Naast de standaard beschikbare plugins kunnen er ook eigen plugins worden ontwikkeld met behulp van de plugin-API. Voor een aantal DUO-specifieke onderdelen zal dit nodig zijn. Daarnaast zullen de volgende standaard plugins nodig zijn:

- IBM WebSphere Application Server Plugin
- IBM WebSphere MQ Plugin
- DB2-database Plugin

Voor het uitvoeren van een deployment moet een zogeheten DAR-bestand (Deployment Archive) worden gemaakt. Dit is eigenlijk een JAR-bestand (Java Archive) met daarin een aangepast manifest met de instructies voor Deployit. Met behulp van dit bestand kunnen de overzettingen vervolgens geautomatiseerd worden uitgevoerd. Het voordeel is dat de configuratie alleen bij de eerste deployment moet worden ingesteld. Hierbij zal de ontwikkelaar die over de juiste kennis beschikt betrokken zijn. Doordat alle deployments met Deployit worden uitgevoerd, wordt bij de eerste deployment direct getest of de configuratie juist werkt. Voor het

overzetten van deze applicatie (in de toekomst), naar welke omgeving dan ook, kan dit bestand worden gebruikt voor de geautomatiseerde overzettingen.

Apache Ant

Voor een automatische deployment naar een WebSphere applicatieserver kan gebruik worden gemaakt van Apache Ant [20]. Dit is een Java-library en commandline tool geschreven in Java. Het doel van de tool is het automatiseren van het bouwproces van software en de verschillende onderdelen, die afhankelijk zijn van elkaar, op de juiste volgorde uitvoeren. Hierbij wordt gewerkt met doelen, waarbij een aantal taken moet worden uitgevoerd om een doel te bereiken. Dit wordt beschreven in een XML-bestand. De tool wordt vooral gebruikt voor Java-applicaties, maar kan ook voor andere programmeertalen worden gebruikt.

WebSphere is geschikt voor gebruik in combinatie met Ant. Hiervoor levert IBM een aantal "Antlibs" mee. Dit zijn Ant-taken die functionaliteiten bieden voor WebSphere. Hierdoor is het onder andere mogelijk om vanuit Ant WSAdmin aan te roepen en om daarmee bijvoorbeeld een Jython-script uit te voeren. Op deze manier kan het hele proces met Ant worden geautomatiseerd.

Ant maakt gebruik van EAR-bestanden (Enterprise Archive). Dit is vergelijkbaar met het DAR-bestand dat bij Deployit gebruikt wordt. Het is in feite een JAR-bestand met daarin naast de applicatie een metadatamap die de benodigde deploymentgegevens bevat.

De tool is opensource en kan kosteloos worden gebruikt. Voor het automatiseren van het gehele buildproces zal echter nog het nodige zelf moeten worden ontwikkeld. Voor WebSphere zijn Ant-taken beschikbaar, maar ook de andere onderdelen (zoals het overzetten van een database) zijn onderdeel van een volledige software deployment.

Ontwerpkeuze

Met de functionaliteiten die Deployit biedt is het mogelijk de volledige overzetting, dus zowel de database als de programmatuur met bijbehorende configuratie, geautomatiseerd uit te voeren. Door het pluginsysteem met bijbehorende API's is het mogelijk (en voor DUO noodzakelijk) om zelf functionaliteiten toe te voegen. Apache Ant is een gratis alternatief, maar er moet veel zelf worden ontwikkeld voordat het te gebruiken is. Voor de DUO-specifieke onderdelen biedt Ant meer flexibiliteit, omdat er geen rekening moet worden gehouden met een plugin-API.

Voor DUO is Deployit het meest geschikt, doordat de tool gemaakt is voor het gehele deploymentproces en over de juiste plugins beschikt. Het is tevens een voordeel dat er al een project loopt dat zich bezighoudt met Deployit. DUO beschikt al over kennis die gebruikt kan worden voor de koppeling met TPM.

5.2 Custom tooling

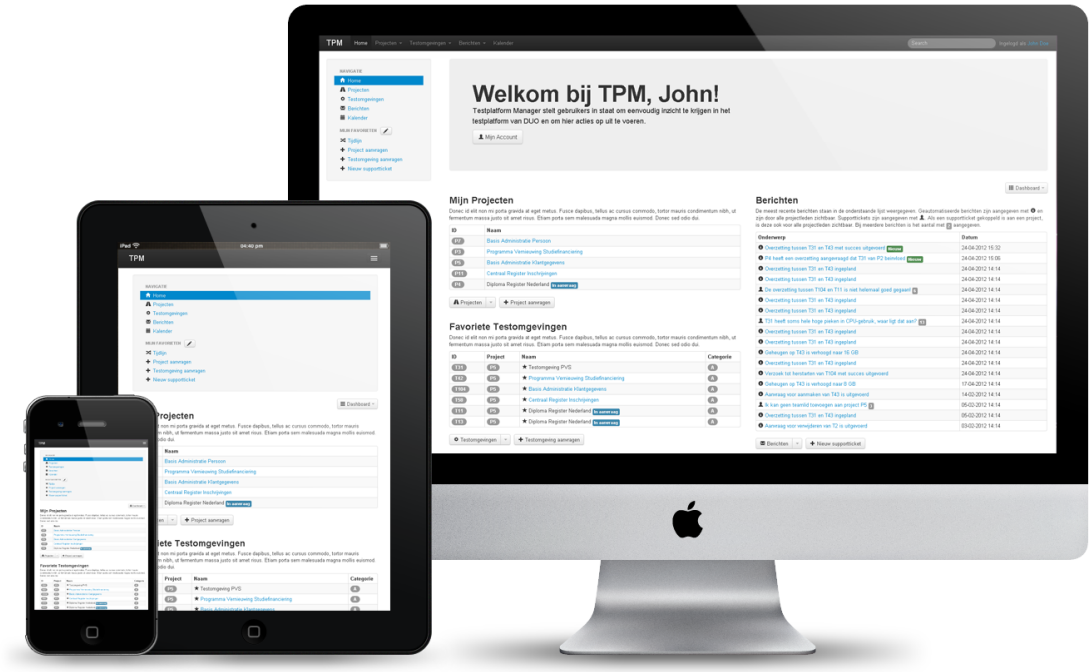
Een aantal belangrijke requirements zijn gedekt door third-party tooling, zoals het automatisch overzetten tussen omgevingen en de onderliggende cloudinfrastructuur. Voor de resterende requirements die softwarematig moeten worden opgelost is custom tooling (maatwerksoftware) nodig.

In de architectuur is rekening gehouden met custom tooling. De maatwerksoftware heeft in dit onderzoek de werktitel "TPM" gekregen. Hoewel de implementatie van TPM buiten de scope van het onderzoek valt, is er wel voor gekozen om een proof-of-concept te maken. Dit is een voorbeeld van hoe TPM er visueel uit zou *kunnen* zien. Het is geen functioneel werkende demo.

In de komende paragrafen is de proof-of-concept van TPM met behulp van screenshots toegelicht. Aan de scriptie is een ⊙ CD toegevoegd met de bronbestanden. Hiermee kan de proof-of-concept op elke PC worden bekeken.

Moderne technieken

Het onderzoek richt zich op een toekomstgericht testplatform. Daar horen ook moderne technieken voor het web bij. In de proof-of-concept is gebruik gemaakt van Twitter Bootstrap [21]. Dit is een moderne gebruikersinterface-framework dat HTML5 Boilerplate, CSS3 en jQuery toepast. Het ondersteunt ook responsive design (figuur 5.2), zodat TPM op meerdere type devices te gebruiken is.

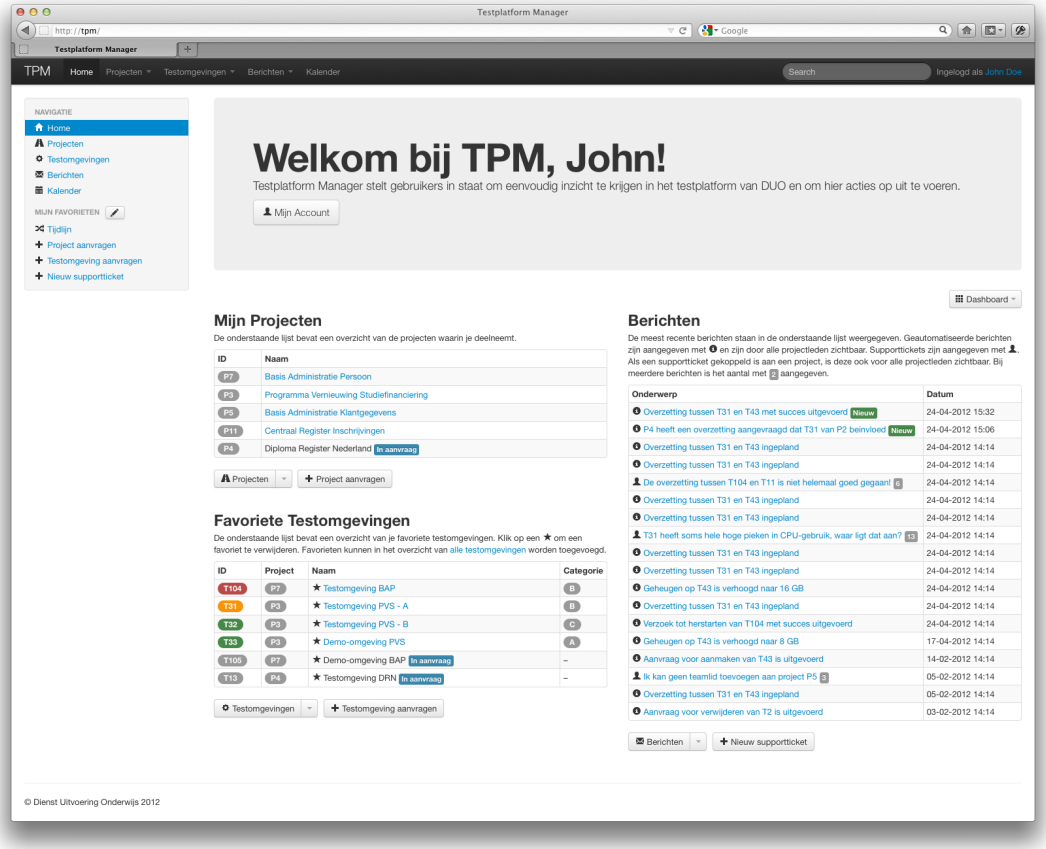


Figuur 5.2: Met behulp van responsive design moet TPM op zowel desktops, notebooks, tablets als mobiele telefoons makkelijk in gebruik zijn.

Hoofdpagina

Een concept van de hoofdpagina van TPM is in figuur 5.3 weergegeven. Aan de boven- en linkerkant van de pagina bevinden zich navigatiemenu's. In het linkermenu kunnen veelgebruikte pagina's op TPM als favoriet worden toegevoegd. De gebruiker kan het zoekveld rechtsboven gebruiken om projecten, testomgevingen, gebruikers, software of andere entiteiten op het testplatform te vinden.

De hoofdpagina bevat een dashboard met widgets. Dit zijn informatieblokken die de gebruiker zelf naar wens kan indelen en configureren. In het voorbeeld zijn de widgets Mijn Projecten, Favoriete Testomgevingen en Berichten weergegeven. In de widgets zijn veelgebruikte acties direct beschikbaar, zoals het aanvragen van een testomgeving of het aanmaken van een supportticket. Met behulp van icoontjes, labels en badges kan de gebruiker snel de status van een entiteit inzien.

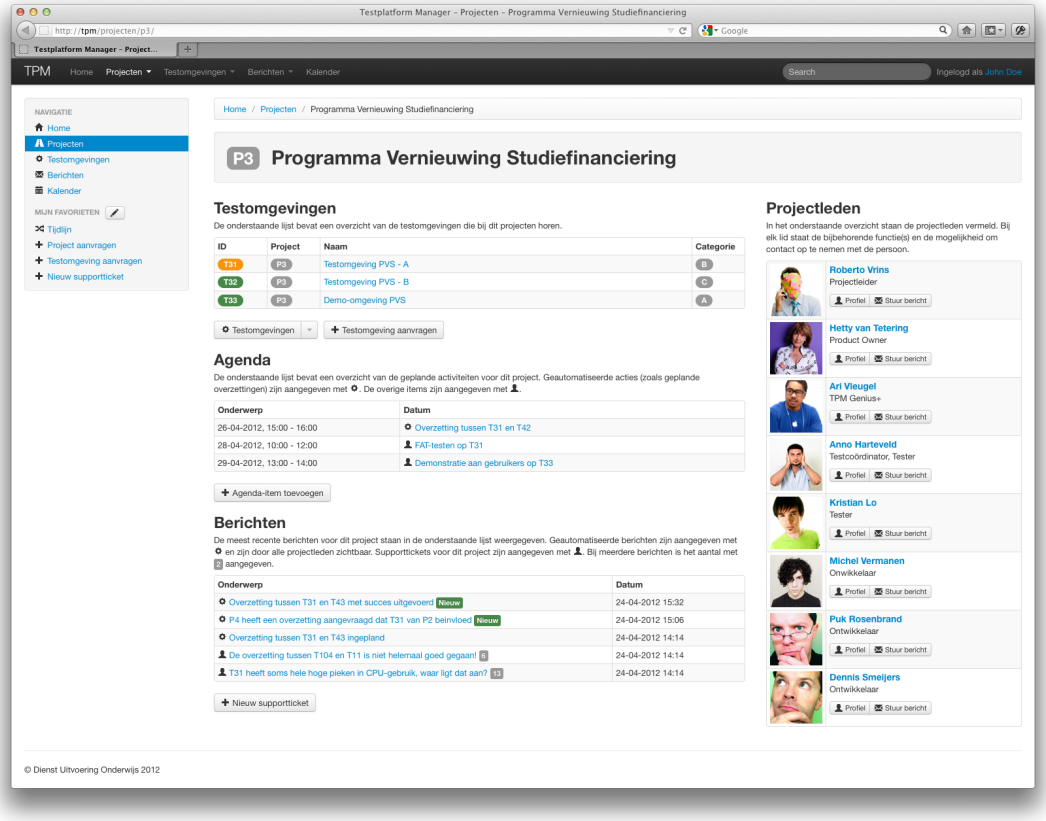


Figuur 5.3: Een concept van de hoofdpagina van TPM. Gebruikers kunnen de pagina met behulp van widgets zelf naar wens indelen.

Projectpagina

Figuur 5.4 bevat een concept van een projectpagina in TPM. De pagina bevat een overzicht van de testomgevingen, agenda-items, berichten en projectleden die bij het project horen. De agenda bevat zowel geautomatiseerde items (zoals

geplande overzettingen) als handmatige afspraken (zoals een functionele test of een demonstratie aan eindgebruikers). Veelgebruikte acties zijn via knoppen snel beschikbaar, zoals het toevoegen van een agenda-item, het sturen van een bericht aan een projectlid of het verwijderen van een testomgeving.

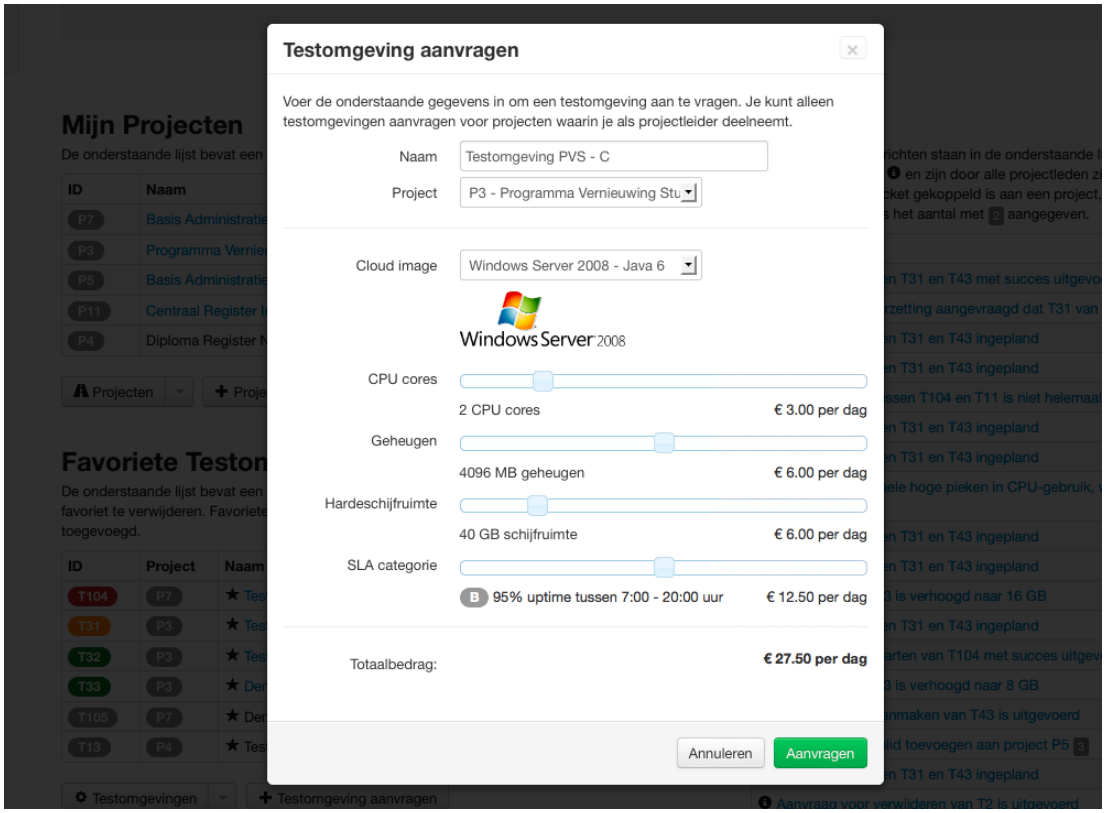


Figuur 5.4: Een concept van een projectpagina in TPM. Als voorbeeld is het project Programma Vernieuwing Studiefinanciering gebruikt.

Testomgeving aanvragen

De requirements bepalen dat testomgevingen eenvoudig aan te vragen moeten zijn. Ook moeten de kosten voor het gebruik van een testomgeving inzichtelijk zijn bij het aanvragen.

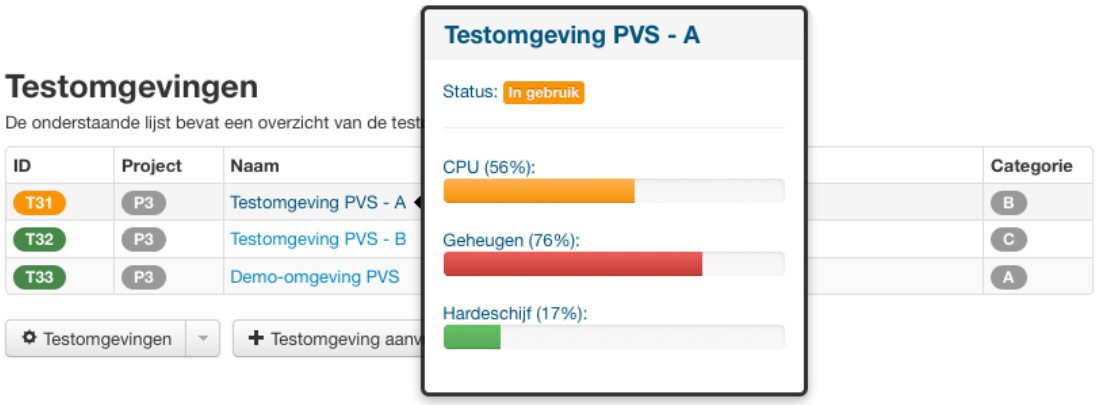
In figuur 5.5 is een concept weergegeven van het aanvraagscherm voor een testomgeving. De naam van de testomgeving en het project waar deze bij hoort moet worden opgegeven. Vervolgens kan een keuze gemaakt worden uit de cloudimages, het aantal CPU-cores, de hoeveelheid geheugen en hardeschijfruimte en de SLA categorie. De kosten worden bij het verslepen van de sliders direct bijgewerkt. Na het aanvragen verschijnt de testomgeving in de Favoriete Testomgevingen widget en kan de status van de aanvraag worden ingezien.



Figuur 5.5: Het aanvragen van een testomgeving in TPM. De gebruiker kan eenvoudig de parameters instellen en de kosten zijn direct inzichtelijk.

Tooltips

Om snel inzicht te kunnen geven in het testplatform, is in de proof-of-concept gebruik gemaakt van tooltips (figuur 5.6). Wanneer in TPM een naam of ID van bijvoorbeeld een testomgeving, project, software of agenda-item voorkomt, wordt er automatisch een tooltip geplaatst met meer relevante informatie.



Figuur 5.6: Tooltips geven snel inzicht in bijvoorbeeld de testomgevingen.

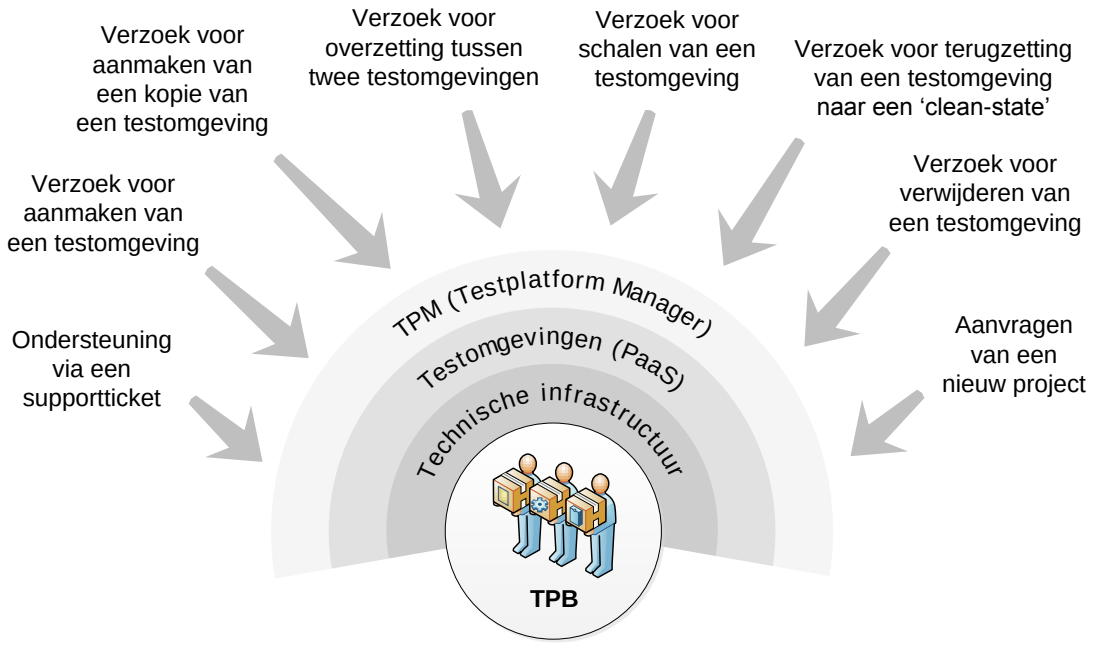
6 Beheer

Het laatste onderdeel van het testplatform omvat het beheer. De architectuur uit hoofdstuk 4, de tooling uit hoofdstuk 5 en de beheersaspecten uit dit hoofdstuk geven samen een volledig beeld van hoe het toekomstgerichte testplatform voor DUO eruit zou moeten zien.

In de eerste paragraaf is beschreven welke partij het beheer van het testplatform voor zijn rekening moet nemen en welke invulling deze partij aan deze rol moet geven. De tweede paragraaf geeft antwoord op de vraag hoe Scrum-projecten het beste gebruik zouden kunnen maken van het onderzochte testplatform.

6.1 Beherende partij

In hoofdstuk 2 is naar voren gekomen dat er geen duidelijkheid bestaat over de verantwoordelijkheid voor de testomgevingen. Voor het toekomstgerichte testplatform moet daarom een nieuw cluster binnen I&E worden opgericht: TPB (Testplatform Beheer). Figuur 6.1 geeft de verantwoordelijkheden van TPB weer en de mogelijke verzoeken van gebruikers aan het testplatform.



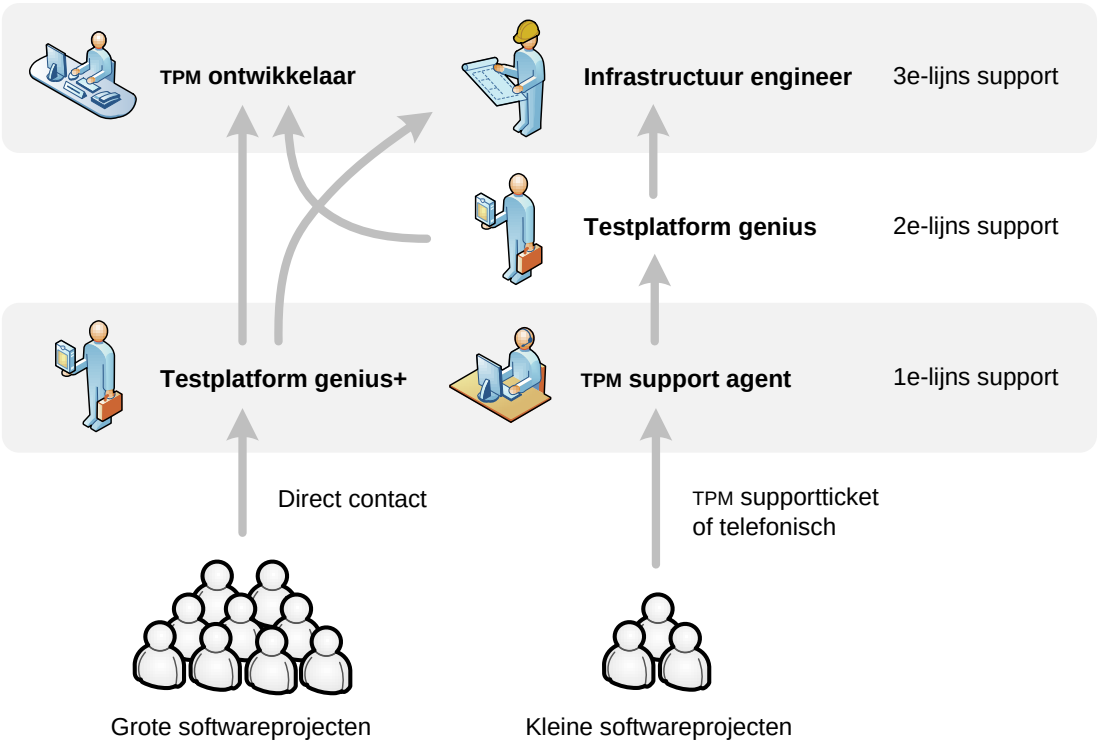
Figuur 6.1: De drie lagen (infrastructuur, PaaS en TPM) waar TPB verantwoordelijk voor moet zijn en mogelijke verzoeken die via TPM binnenkomen.

De redenen om het beheer van het testplatform aan een nieuw cluster bij I&E toe te wijzen zijn:

- Het beheer kan het beste door één partij worden uitgevoerd om zo een wildgroei van communicatiekanalen en kennisverspreiding tussen afdelingen en de daarbij horende vertragingen te voorkomen.
- Omdat het testplatform de technische infrastructuur, de testomgevingen en het TPM softwaresysteem omvat, zal het beheer goed bij een cluster binnen I&E passen (zie paragraaf 1.1.1 voor de afdelingomschrijvingen van DUO).
- Met de cluster TPB wordt een identiteit gecreëerd die de rest van de organisatie en het managementteam duidelijkheid geeft over de beherende partij van het testplatform. Door in het organogram van DUO te staan zal het managementteam de belangen van het testplatform eerder begrijpen.

6.1.1 Rolverdeling

Voor het goed kunnen beheren van het testplatform zijn veel vaardigheden vereist. Deze vaardigheden kunnen moeilijk door eenzelfde persoon worden uitgevoerd. Het is daarom belangrijk een duidelijke rolverdeling te hebben. Figuur 6.2 bevat een overzicht van de rollen die voor het beheer van TPM nodig zijn en toont hoe de communicatie tussen de gebruiker en deze rollen verloopt.



Figuur 6.2: De verschillende rollen die voor TPM nodig zijn en de communicatieroutes vanaf de gebruiker van het testplatform tot aan 3e-lijns support.

Voor softwareprojecten is TPB bereikbaar via 1e-lijns support. Bij kleine projecten verloopt dit via een TPM support agent. Complexe problemen kunnen naar 2e- of 3e-lijns support doorgeschakeld worden. Grotere projecten kunnen ervoor kiezen om on-site ondersteuning (een genius+) voor het testplatform in te huren. Tabel 6.1 bevat een gedetaileerde omschrijving van de verschillende rollen.

Rol	Vaardigheden en verantwoordelijkheden
TPM support agent	Verwerkt binnenkomende supportaanvragen via TPM (zie figuur 6.1) of de telefoon. Een support agent moet goede kennis van TPM en het testplatform hebben en moet kleine projecten kunnen adviseren over het inrichten van testomgevingen. Wanneer een vraag niet door een support agent kan worden beantwoord, dient deze te worden overgenomen door een testplatform genius.
Testplatform genius	Een testplatform genius is een expert op het gebied van TPM en het testplatform. Een genius heeft veel bevoegdheden, zoals het kunnen aanmaken van nieuwe cloudimages en het direct kunnen schalen van testomgevingen. Een genius blijft continu op de hoogte van de ontwikkelingen op het testplatform en in het algemeen van lopende Scrum-projecten bij DUO.
Testplatform genius+	Een testplatform genius+ heeft dezelfde vaardigheden als een genius. Daarbovenop is een genius+ on-site inzetbaar (fysiek aanwezig) bij grotere projecten. Een genius+ moet in grote lijnen de architectuur, afhankelijkheden en relaties van de project(en) kennen waarvoor hij is ingezet. Dit om de inrichting van de testomgevingen voor het project zo optimaal mogelijk te kunnen doen en om verzoeken en problemen direct aan te kunnen pakken.
Infrastructuur engineer	De testomgevingen draaien als virtuele instanties op een private cloud. Het technisch beheer van de cloud wordt uitgevoerd door infrastructuur engineers. Een engineer is een expert op het gebied van cloud computing en IBM Workload Deployer. Het is de verantwoordelijkheid van de engineer om het testplatform draaiende te houden. TPM helpt hierbij met actieve monitoring.
TPM ontwikkelaar	Een TPM ontwikkelaar is verantwoordelijk voor het oplossen van bugs, refactoren van code, testen van TPM en toevoegen van nieuwe functionaliteiten aan TPM.

Tabel 6.1: De vaardigheden en verantwoordelijkheden die bij de rollen horen.

Bij de afdeling I&E wordt geen software ontwikkeld, waardoor de rol van TPM ontwikkelaar het beste door het Softwarehuis of een externe partij kan worden uitgevoerd. Hoofdstuk 7 (systeemevolutie) gaat hier verder op in. De andere rollen dienen door de cluster TPB te worden uitgevoerd. Een aantal ondersteunende rollen ontbreken in het overzicht, zoals bijvoorbeeld het management van de cluster.

6.1.2 Service Level Management

Het is belangrijk afspraken te maken over de kwaliteit van de diensten die TPB zal leveren naar de gebruikers van het testplatform. Service Level Management is het proces van onderhandelen, het definiëren, meten, beheren en verbeteren van de kwaliteit van de (IT-)dienstverlening. Hierbij moet een balans worden gevonden tussen de kwaliteit, de vraag, de gebruiksvriendelijkheid en de kosten [22] [23].

In SLA's (Service Level Agreements) kunnen deze afspraken worden opgenomen. Het voordeel van een dergelijke overeenkomst is dat de afspraken op papier staan en dat een gebruiker een bepaalde kwaliteit van een dienst kan verwachten. Indien hier niet aan wordt voldaan, moeten er vastgestelde sancties voor TPB tegenover staan. Een onafhankelijk orgaan dient de audits uit te voeren, zodat er geen belangenverstrengeling ontstaat.

Definitie van SLA

SLA staat voor Service Level Agreement. Dit is een overeenkomst tussen de klant en de IT dienstenleverancier waarin overeengekomen kwaliteitsniveaus van de dienstverlening gedetailleerd en meetbaar zijn vastgelegd. Dit is beschreven met niet-technische termen, zodat het voor beide partijen duidelijk is wat er kan worden verwacht.

Voordat de SLA's kunnen worden opgesteld, moeten de te leveren diensten geïdentificeerd zijn en beschreven staan in een dienstencatalogus. Een aantal voorbeelden van diensten die de cluster TPB aan gebruikers kan leveren:

- leveren van testomgevingen
- binnenkomende (support)aanvragen in TPM beantwoorden en uitvoeren
- advies geven over de inrichting van testomgevingen
- on-site ondersteuning geven.

In een SLA kunnen de volgende gegevens worden opgenomen:

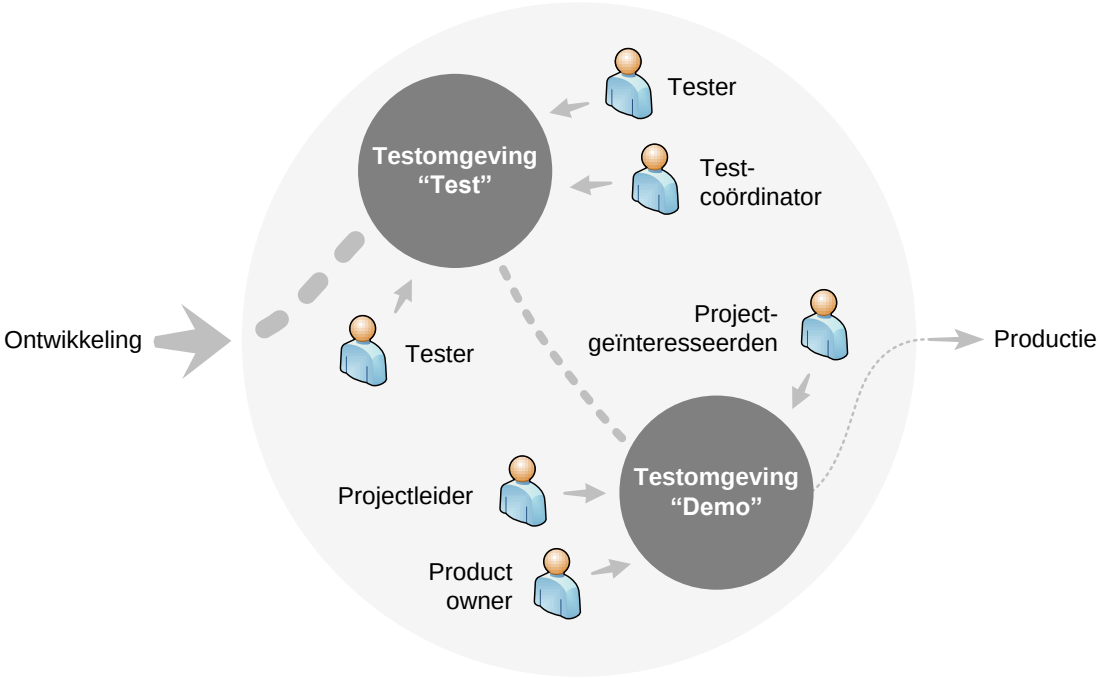
- de te leveren diensten
- servicetijden
- reactietijden
- beschikbaarheid
- performance
- minimale en maximale afname
- rapportages
- sancties
- hoe gaat men om met wijzigingen binnen een SLA
- kostenbeheersing
- monitoring, metingen en audits.

Voor de projecten die bij DUO worden uitgevoerd is het belangrijk dat is vastgelegd wat er van de dienstverlening kan worden verwacht. Het testplatform is een belangrijke voorziening die niet tot vertraging mag leiden. Met behulp van verschillende SLA's en daarmee een verschil in kosten kan een hogere kwaliteit worden geleverd aan projecten waar beschikbaarheid meer van belang is.

6.2 Scrum en testomgevingen

Momenteel wordt er bij enkele projecten al gebruik gemaakt van de Scrum ontwikkelmethodiek. In de toekomst wil DUO deze Agile ontwikkelmethodiek nog breder gaan toepassen. In deze paragraaf is beschreven hoe de testomgevingen hiervoor zouden kunnen worden ingericht. Dit maakt duidelijk waarom het toekomstgerichte testplatform hier geschikt voor is. De Scrum ontwikkelmethodiek is verder toegelicht in bijlage C.

Met het toekomstige testplatform is er niet meer de verplichte (op OTAP gebaseerde) indeling van testomgevingen, want met TPM kunnen de testomgevingen (tussen de ontwikkel- en productieomgeving) per project naar wens worden ingedeeld. In figuur 6.3 is een mogelijke inrichting voor gebruik met de Scrum ontwikkelmethodiek weergegeven.



Figuur 6.3: Een mogelijke inrichting van de testomgevingen voor gebruik met de Scrum ontwikkelmethodiek. Hoe dikker de stippellijn, hoe vaker een overzetting plaatsvindt.

Er is gekozen voor twee testomgevingen: “Test” en “Demo”. Zoals de naam al aangeeft wordt de eerste testomgeving gebruikt voor het uitvoeren van verschillende soorten tests. Als de ene test is afgerond kan een ander teamlid van dezelfde testomgeving gebruik maken om de volgende soort test uit te voeren. Een andere mogelijkheid is dat hetzelfde teamlid meerdere rollen heeft en daardoor verschillende soorten tests kan uitvoeren. Het is gebruikelijk dat de teamleden in een Scrumteam meerdere rollen hebben.

De tweede testomgeving wordt gebruikt voor het geven van demonstraties aan de product owner en andere geïnteresseerden. Het is belangrijk dat de omgeving tijdens de demo beschikbaar is. Het is daarom verstandig om voor een SLA te kiezen met een hogere beschikbaarheid. Door af te spreken dat de demonstratieomgeving alleen demo-rijpe versies mag bevatten, is op ieder moment een stabiele versie beschikbaar. Demo-rijp wil zeggen dat de software voldoende getest, zodat deze als demo gebruikt kan worden. Als de software gereed is voor productie kan deze omgeving ook gebruikt worden voor het uitvoeren van de pre-productietests, zoals de hack- en performancetest.

Een project bestaat uit meerdere sprints, waarbij telkens van dezelfde testomgevingen gebruik wordt gemaakt. In figuur 6.3 geeft een dikkere stippellijn aan dat er vaker overzettingen plaatsvinden. Hierbij is te zien dat er vanuit de ontwikkelomgeving vaker nieuwe versies naar de “Test” omgeving worden overgezet dan dat er overzettingen naar de “Demo” omgeving uitgevoerd worden, namelijk minstens één keer per sprint. Pas op het eind van een project wordt een versie in productie genomen, wat de dunne lijn naar productie duidelijk maakt. Verder moet er worden opgemerkt dat de lijnen geen verplichte route aangeven, zoals dat in de huidige situatie het geval was. Het is mogelijk om vanaf een willekeurige omgeving een overzetting te doen naar een willekeurige andere omgeving. Door deze opzet neemt het aantal benodigde omgevingen en daarmee het aantal overzettingen af, wat het proces minder complex maakt.

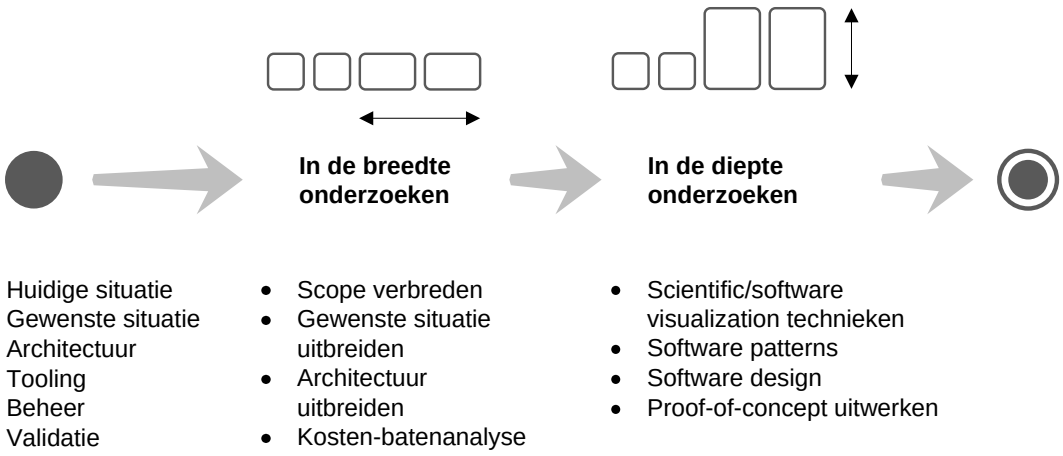
7 Systeemevolutie

Het laatste deel van dit onderzoek omvat de evolutie van het onderzochte testplatform. Het is belangrijk om de mogelijke vervolgstappen voor dit onderzoek in kaart te brengen. Dat vergroot de kans dat de onderzoeksresultaten worden gebruikt en geeft de verschillende stakeholders een goed toekomstperspectief van het project. In de komende paragrafen is in een chronologische volgorde de systeemevolutie van het testplatform uitgestippeld.

7.1 Vervolgonderzoek

De eerste stap die ondernomen moet worden is een vervolgonderzoek. Er is bij dit project onderzoek gedaan naar de huidige situatie, gewenste situatie (in de vorm van requirements), architectuur, tooling en beheersaspecten voor een toekomstgericht testplatform. De onderzoeksresultaten hiervan zijn ook gevalideerd.

In een vervolgonderzoek kan gekeken worden naar requirements en aanvullingen op de architectuur voor punten die niet binnen de scope van dit afstudeeronderzoek vallen. Ook kan er onderzoek gedaan worden naar software patterns die toepasbaar zijn in de architectuur en naar het ontwerp (design) van TPM. De proof-of-concept zou verder uitgebreid kunnen worden. Figuur 7.1 geeft de stappen voor een vervolgonderzoek visueel weer. Om herhalende problemen bij het onderzoeken te voorkomen is een leerpuntenrapport opgesteld (zie bijlage F).



Figuur 7.1: De stappen voor een vervolgonderzoek. Er kan eerst in de breedte verder onderzoek gedaan worden, om vervolgens de diepte in te gaan.

7.2 Realisatie

Na het vervolgonderzoek kan het testplatform worden gerealiseerd. In deze paragraaf is beschreven hoe dit gefaseerd kan worden en welke partijen het kunnen uitvoeren.

7.2.1 Gefaseerd invoeren

In de voorgaande hoofdstukken is duidelijk geworden dat een nieuw testplatform voor DUO een noodzakelijke verandering is. Uit onderzoek van de Algemene Rekenkamer over ICT-projecten bij de overheid [24] blijkt dat in veel gevallen onderschat wordt hoeveel impact de business en ICT op elkaar hebben. Daarnaast ontstaan de meeste problemen bij grote projecten; deze worden gedurende het project duurder en lopen in veel gevallen vertraging op. Om te zorgen dat het nieuwe testplatform succesvol wordt gerealiseerd en de medewerkers stapsgewijs overstappen naar het nieuwe systeem is gefaseerd invoeren belangrijk.

Door het gefaseerd invoeren wordt het project opgesplitst in kleinere onderdelen en kan de werkwijze bij de organisatie stapsgewijs aan het nieuwe systeem worden aangepast. In tabel 7.1 staat een overzicht van de mogelijke fasen. Afhankelijk van het vervolgonderzoek kan het zijn dat er enkele fasen veranderen, worden toegevoegd of de volgorde wordt aangepast.

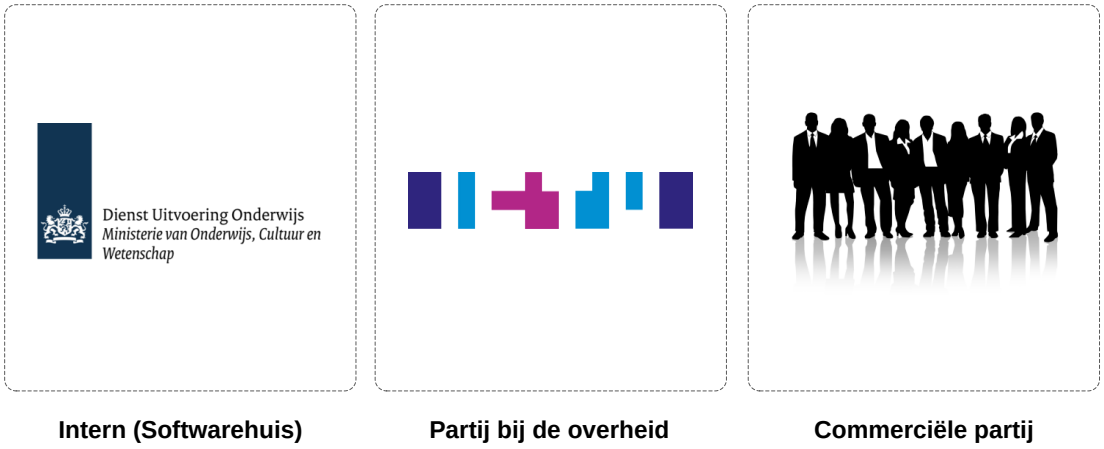
Fase	Omschrijving
Basis	In deze fase zal de basis voor TPM worden gerealiseerd. Deze fase bevat de functionaliteiten om gebruikers en projecten toe te voegen en het toewijzen van rollen aan gebruikers. De basis is generiek opgebouwd, zodat (nieuwe) functionaliteiten eenvoudig kunnen worden toegevoegd. Er wordt ook een start gemaakt met de cluster TPB.
Private cloud	In deze fase wordt de private cloud opgezet en wordt het aanvragen van testomgevingen in TPM gerealiseerd. Losse formulieren per e-mail komen daarmee te vervallen. De cluster TPB moet in een operationeel stadium zijn en Service Level Management hebben geïntegreerd in de processen. Alle TPB rollen (zoals de genius) moeten getraind zijn.
Geautomatiseerd overzetten	In deze fase wordt het geautomatiseerd overzetten met Deployit gerealiseerd. Aanvragen voor overzettingen worden gedaan in TPM waardoor vrachtbrieven per e-mail niet langer nodig zijn.
Afhankelijkheden	In deze fase worden afhankelijkheden in software gedetecteerd en in een TPM service geregistreerd. De afhankelijkheden zijn daarmee zichtbaar in de overzichten van TPM en bij het uitvoeren van acties worden afhankelijke projecten automatisch op de hoogte gebracht.
Bevindingen	In deze fase wordt de koppeling met bevindingen gerealiseerd. De tijdlijn van een project kan hiermee worden uitgebreid.

Tabel 7.1: De opbouwende fasen waarin de realisatie kan worden uitgevoerd.

Door het gefaseerd invoeren van het nieuwe testplatform kunnen medewerkers geleidelijk wennen aan de nieuwe situatie, waarbij steeds meer via het centrale systeem wordt geregeld. Het werken met losse bestanden en e-mail (zoals dat in de huidige situatie het geval is) heeft als voordeel dat dit goed te combineren is met het gefaseerd invoeren van het nieuwe systeem. Stapsgewijs zullen oude documenten niet meer nodig zijn, maar worden deze gegevens in het centrale systeem opgenomen of worden deze automatisch gegenereerd. Medewerkers zullen aan de nieuwe manier van werken gewend raken, de voordelen van de nieuwe functionaliteiten inzien en gemakkelijker afstand van de oude documenten doen.

7.2.2 Realiserende partij

Voor de realisatie van het nieuwe testplatform kan worden gekozen voor verschillende partijen. Figuur 7.2 geeft een overzicht van de mogelijke partijen. In deze paragraaf is per blok een toelichting gegeven.



Figuur 7.2: Mogelijke partijen die de ontwikkeling en implementatie van het testplatform kunnen uitvoeren.

Intern

Het Softwarehuis bij DUO zou de realisatie van het testplatform op zich kunnen nemen. Deze afdeling beschikt over voldoende kennis en heeft ervaring met grote projecten. Dit project zal naast de huidige activiteiten moeten worden uitgevoerd, waardoor er mogelijk extra capaciteit nodig is.

Voor de realisatie moet een team komen dat zich fulltime met dit project bezig kan houden. Omdat het testplatform gefaseerd zal worden ingevoerd kan het voordelen hebben dit intern te ontwikkelen. De medewerkers van het Softwarehuis hebben kennis van de huidige situatie en kunnen ondersteuning bieden bij het in gebruik nemen van het nieuwe systeem. Eventuele problemen kunnen intern waarschijnlijk tijdiger worden opgelost. Bovendien beschikt DUO dan over alle inhoudelijke kennis met betrekking tot de werking van het testplatform. De

voorkennis van de medewerkers van het Softwarehuis kan echter ook als nadeel worden gezien, omdat ze te veel worden beïnvloed door de huidige manier van werken.

Partij bij de overheid

ICTU staat voor ICT Uitvoeringsorganisatie en is een instelling van en voor overheden. De organisatie is in 2001 in het leven geroepen om uitsluitend opdrachten voor overheden uit te voeren. Het werkveld is de digitale overheid, waarbij ze zowel zorgt voor het verbeteren van de werkprocessen bij overheden als de dienstverlening aan de maatschappij. Dit alles met als doel een verbonden overheid die beter presteert voor burgers en bedrijven.

Het nieuwe testplatform is een duidelijk voorbeeld van het verbeteren van het werkproces bij DUO. Hierdoor zou ICTU een geschikte partij kunnen zijn om de realisatie van het testplatform aan uit te besteden. Doordat ICTU net als DUO een overheidsorganisatie zonder winstoogmerk is, maakt dat de samenwerking waarschijnlijk eenvoudiger dan met een commerciële partij, waarbij het commerciële belang altijd een rol speelt. Dit is een voordeel bij het bieden van ondersteuning en om in de toekomst het testplatform uit te breiden. Het is het misschien ook mogelijk een samenwerking aan te gaan waarbij ICTU de basis ontwikkelt en het Softwarehuis in een later stadium de ontwikkeling en ondersteuning overneemt. Doordat ICTU ervaringen heeft met projecten voor andere organisaties en niet beïnvloed wordt door de huidige manier van werken bij DUO, kunnen goede ideeën naar voren komen.

Commerciële partij

Naast bovenstaande mogelijkheden kan er ook voor worden gekozen om het project uit te besteden aan een commerciële partij. Welke organisatie het meest geschikt is zal van verschillende factoren afhangen:

- Is het belangrijk dat de organisatie een kantoor in de omgeving van Groningen heeft?
- Gaat de voorkeur uit naar een groot of klein bedrijf?
- Welke ervaringen heeft de overheid al met een bepaalde organisatie?
- Wat zou het project gaan kosten bij de verschillende organisaties?

Naast deze factoren moet er worden vastgelegd hoe de samenwerking in de toekomst verder gaat. In hoeverre kan er (tijdig) support geleverd worden door de partij die de software heeft ontwikkeld? En in hoeverre kan het Softwarehuis deze taak op zich nemen? Afhankelijk van de antwoorden op deze vragen kan worden bepaald of de uitbesteding aan een commercieel bedrijf de juiste keuze is.

7.3 Rijksbreed toepassen

In de verre toekomst kan het testplatform ook bij andere organisaties binnen de rijksoverheid worden toegepast. Figuur 7.3 bevat een overzicht met voorbeelden van overheidsinstanties die zelf software ontwikkelen en baat zouden kunnen hebben bij het onderzochte testplatform.

Figuur 7.3: Mogelijke partijen binnen de rijksoverheid waarvoor het testplatform interessant kan zijn.

Naast het rijksbreed toepassen van het testplatform kan in de toekomst de cloud worden uitgebreid tot een gesloten Rijkscloud. Dit kan dan worden gezien als een community cloud (zie paragraaf 4.3) die door de verschillende partijen binnen de overheid kan worden gebruikt. Op deze manier kan nog efficiënter van resources gebruik worden gemaakt. De term Rijkscloud is in de Tweede Kamer al eens ter sprake geweest [25], waarbij al snel reacties ontstonden dat dit een te groot ICT-project zou worden. Zoals paragraaf 7.2.1 al aangeeft is gefaseerd invoeren belangrijk. De Rijkscloud is dan ook een toekomstplan. Eerst moet met een private cloud worden aangetoond dat er voldoende kennis is en dat een cloud op de juiste manier wordt toegepast.

8 Validatie

In de voorgaande hoofdstukken is onderzoek gedaan naar een toekomstgericht testplatform voor DUO. Het is belangrijk de onderzoeksresultaten te valideren bij de stakeholders van het onderzoek. Mogelijke knelpunten in het ontwerp kunnen daarmee worden geïdentificeerd. Het zorgt er ook voor dat de stakeholders zich bij het project betrokken voelen, op de hoogte zijn van de onderzoeksresultaten en het eindproduct beter bij hen zal aansluiten. Dit hoofdstuk behandelt de methode waarop de validatie is uitgevoerd en de resultaten hiervan.

8.1 Methode

De validatie van de onderzoeksresultaten is met behulp van gesprekken uitgevoerd. Tijdens de gesprekken zijn de resultaten van het onderzoek uitgelegd met behulp van de figuren uit de scriptie. De proof-of-concept uit paragraaf 5.2 en de scenario's uit paragraaf 4.5 zijn gebruikt om de stakeholders een zo duidelijk mogelijk beeld te geven van het toekomstgerichte testplatform. Tabel 8.1 bevat een overzicht van de stakeholders waarmee een validatiegesprek is gehouden.

	Afdeling of cluster	**Persoon**	**Functie**
1.	Softwarehuis	Andries Mesken	Senior programmeur
		Marko Ketellapper	Senior functioneel tester
2.	Projectmanagement	Robbert Jan van Meenen	Projectmanager
3.	ICT-Demand	Edwin Huijser	Procesbeheerder
4.	ICT-Regie	Jeroen Smit	Projectleider
5.	Softwarehuis	Piet de Haan	Testnavigator
		Gerd van den Berg	Testnavigator
		Marten de Vries	Testnavigator
6.	Software Control	Harke Wijnsma	Technisch objectbeheerder
7.	NT-Beheer	Peter Bloed	Beheerder infrastructuur

Tabel 8.1: Stakeholders van het onderzoek waarmee een validatiegesprek is gehouden. In bijlage B staan deze ook als gesprekken vermeld.

De afdelingen en functies van deze personen komen grotendeels overeen met de geïdentificeerde stakeholders van het onderzoek uit paragraaf 3.1. Dit voorkomt dat mogelijke bevindingen van een stakeholder over het hoofd worden gezien en daarmee de validatie niet volledig wordt uitgevoerd. De resultaten van de validatie staan in de volgende paragraaf beschreven.

8.2 Resultaten

Deze paragraaf bevat de resultaten van de validatiegesprekken. In alle gesprekken gaven de geïnterviewden aan dat de oplossing realistisch is en in een grote mate zal bijdragen aan de efficiëntie van het werken. Het toekomstgerichte testplatform zal volgens de geïnterviewden veel van de huidige problemen wegnemen en ervoor zorgen dat de stakeholders zich weer op hun kerntaken kunnen richten. Tijdens de gesprekken kwamen ook nuttige feedbackpunten naar voren. In de komende paragrafen staan de resultaten hiervan per onderwerp verwerkt.

Onderzoek naar ketens en afhankelijkheden

De ketens en afhankelijkheden die bij de combinatie van Java en een SOA ontstaan vormen een uitdaging bij het testen. In TPM is daar rekening mee gehouden door softwarecomponenten (zoals bestanden, klassen en functies) en de relaties daartussen te registreren en inzichtelijk te maken. In het vervolgonderzoek moet bekeken worden hoe software visualisatietechnieken bij kunnen dragen aan het inzichtelijk maken van de ketens en afhankelijkheden. De geïnterviewden benadrukken dat dit een heel belangrijk punt is voor het vervolgonderzoek. Het is een complex onderwerp dat volgens sommigen zelfs een eigen onderzoek waard is.

Scope van het project uitbreiden, of niet?

Tijdens de validatiegesprekken kwam geen uniform antwoord op de vraag of de scope van het project moet worden uitgebreid. Volgens sommigen moeten de legacystemen meegenomen worden in het toekomstgerichte testplatform. Veel Java-software van DUO maakt gebruik van legacysystemen. Bij het testen is het belangrijk om rekening te houden met de omgevingen op het AS/400-platform en de relaties met deze omgevingen. Niet iedereen is het hier mee eens. Een aantal vinden dat het testplatform zich moet focussen op de toekomst en geen rekening hoeft te houden met de legacysystemen, die ooit zullen verdwijnen.

De keuze voor Deployit opnieuw onderzoeken

In het onderzoek is voor het uitvoeren van overzettingen tussen omgevingen gekozen voor de third-party tool Deployit. Een aantal geïnterviewden plaatsten vraagtekens bij deze beslissing. De overzettingen tussen Java-omgevingen zijn zo specifiek voor DUO dat het misschien loont om hier zelf software voor te ontwikkelen. De voorstanders van deze keuze vinden dat er teveel gewijzigd moet worden aan Deployit voordat het aan de wensen voldoet. Ook is het de vraag of toekomstige updates van Deployit wel zullen werken bij zoveel aanpassingen. De ontwikkelkosten voor een eigen overzettool zullen misschien lager uitvallen dan de licentie-, onderhouds- en ondersteuningskosten van Deployit.

Aandacht besteden aan de cultuurslag

De manier van werken bij het toekomstgerichte testplatform zal anders zijn dan de huidige situatie. Het beheer wordt door één partij uitgevoerd, er zijn nieuwe rollen

binnen deze partij nodig, de testomgevingen zijn anders ingericht, er wordt met SLA's gewerkt en er is één centraal systeem. Zo zijn er nog meer voorbeelden die bijdragen aan een cultuurslag binnen de organisatie. Een aantal geïnterviewden geven aan dat dit niet onderschat moet worden.

Testbevindingen niet in TPM registreren

Voor de realisatie van het testplatform is gekozen voor gefaseerd invoeren (zie paragraaf 7.2.1). Eén van de laatste stappen is het toevoegen van testbevindingen aan TPM. In de ERD is de entiteit testbevinding ook zichtbaar (zie figuur 4.7). Tijdens twee validatiegesprekken werden vraagtekens geplaatst bij de keuze voor het toevoegen van testbevindingen. Het is de vraag of de registratie hiervan wel aansluit bij het doel van TPM en of dit niet het beter in een ander systeem kan gebeuren. Dit zou in het vervolgonderzoek kunnen worden meegenomen.

Koppeling leggen met ITSM

ITSM is een belangrijke tool die bij DUO de processen met betrekking tot IT service management faciliteert. Tijdens een aantal validatiegesprekken kwam naar voren dat de koppeling tussen TPM en ITSM wordt gemist. Deze koppeling zou belangrijk zijn voor met name de procesgeoriënteerde gebruikers van het testplatform.

Licentiekosten voor virtuele testomgevingen

In het onderzoek is geen aandacht besteed aan de licentiekosten die komen kijken bij virtuele testomgevingen. Wat moet er exact worden betaald voor de licenties van bijvoorbeeld Windows en de WebSphere applicatieservers? Is het voor de licentiekosten niet beter om te schalen in de hoeveelheid gigahertz in plaats van het aantal CPU-cores? Hier zou volgens een geïnterviewde onderzoek naar moeten worden gedaan.

Motivering voor maatwerksoftware versterken

Tijdens het onderzoek is gekozen voor een combinatie van third-party en custom tooling (maatwerksoftware). De maatwerksoftware heeft de werktitel TPM gekregen. Bij één validatiegesprek werd aangegeven dat de motivering voor de maatwerksoftware uitgebreider kan en er sterkere argumenten kunnen worden gegeven. Bij DUO geldt de policy dat zoveel mogelijk voor third-party oplossingen moet worden gekozen. De ontwikkelkosten die bij maatwerksoftware komen kijken moeten door managers worden begrepen. Het risico is dat het project anders mogelijk niet van de grond komt.

Werken met kosten: de gevolgen van een leeg potje moeten voelbaar zijn

Het idee van werken met kosten zal volgens een geïnterviewde alleen werken als er consequenties aan vastzitten. Werknemers zouden op een bepaalde manier de gevolgen moeten voelen wanneer het geld bijna op is. Als op dat moment gebruik wordt gemaakt van een ander potje met geld, zal het systeem van interne kosten niet werken. Dit punt sluit aan bij de cultuurslag die nodig is bij DUO.

De testnavigator: een vergeten stakeholder

De testnavigatoren zijn onderdeel van de vakgroep Test van het Softwarehuis en geven technische ondersteuning aan de functionele testers. Een aantal geïnterviewden gaf aan dat de testnavigator onterecht niet vermeld is als stakeholder in de gewenste situatie (hoofdstuk 3). De testnavigatoren zouden bepaalde wensen kunnen hebben die de requirements en daarmee de architectuur, tooling en het beheer kunnen beïnvloeden.

Overzicht van stubs toevoegen aan TPM

Bij het uitvoeren van tests zijn er vaak afhankelijkheden aanwezig. Tijdens één van de gesprekken kwam het idee naar voren om aan TPM een overzicht van beschikbare stubs toe te voegen. Een stub reageert net als een echt onderdeel van een systeem, maar levert slechts een beperkte functionaliteit. Niet in alle gevallen is de echte applicatie noodzakelijk. Door een overzicht te hebben van alle reeds beschikbare stubs kan hier efficiënter gebruik van worden gemaakt en zijn minder kopieën van volledige systemen nodig.

9 Conclusie

Dit hoofdstuk bevat de conclusies van het onderzoek en geeft daarmee antwoord op de onderzoeksvraag: *"Hoe kan het testplatform van* DUO *het beste worden ingericht, zodat het voldoet aan de toekomstvisie waarin Java, een* SOA *en de Scrum ontwikkelmethodiek centraal staan?"*

Uit de analyse van de huidige situatie kan worden geconcludeerd dat de huidige manier van werken niet goed aansluit bij Java en de Scrum ontwikkelmethodiek. Er wordt te weinig gebruik gemaakt van automatisering, waardoor veel tijd verloren gaat aan het oplossen van problemen die ontstaat door menselijke fouten. Testers werken elkaar soms tegen, omdat er door de SOA meer afhankelijkheden zijn waarvan het overzicht mist. Bovendien is het is niet verstandig dat de verantwoordelijkheid voor de testomgevingen verdeeld is over meerdere afdelingen. De wensen van de verschillende stakeholders sluiten goed aan bij bovenstaande conclusies. Daarnaast is de beschikbaarheid van testomgevingen belangrijk en dient het toekomstgerichte testplatform gebruiksvriendelijk te zijn.

Om de problemen op te lossen en aan de wensen te voldoen moet er een centraal systeem komen. Doordat de requirements op veel punten specifiek zijn is een deel maatwerksoftware noodzakelijk. De maatwerksoftware is generiek opgebouwd, zodat eenvoudig nieuwe functionaliteiten kunnen worden toegevoegd. Een deel van de requirements moet met third-party tooling worden opgelost. Er zijn oplossingen op de markt aanwezig en hiermee worden hoge ontwikkelkosten voorkomen. De maatwerksoftware ligt als schil om de third-party tooling heen.

Voor schaalbare en beheerbare testomgevingen is het inrichten van een private cloud in het eigen datacenter de beste keuze. Voor de realisatie van deze PaaS oplossing is IBM Workload Deployer in combinatie met VMware ESX het meest geschikt. Met de aanschaf van Deployit heeft DUO een goede keuze gemaakt als oplossing voor het automatisch overzetten tussen testomgevingen. Wel moet worden bekeken of het configureren van Deployit niet meer tijd kost dan het ontwikkelen van een eigen overzettool.

Voor het beheer moet één partij verantwoordelijk zijn. Een nieuwe cluster binnen I&E is daarbij de meest logische keuze. Om de beschikbaarheid van de diensten te garanderen en te zorgen dat de interne dienstverlening van hoge kwaliteit is, moet met SLA's worden gewerkt. De beherende partij moet de afnemers van de diensten als klant zien. Projecten kunnen zelf bepalen hoeveel en welke soorten testomgevingen er nodig zijn. Door in eenzelfde testomgeving meerdere soorten

tests uit te voeren en kosten direct inzichtelijk te maken moet worden gezorgd dat er efficiënter met testomgevingen wordt omgegaan. Door deze nieuwe manier van werken zal de cultuur binnen DUO moeten veranderen.

Vanwege de breedte van dit onderzoek is de gegeven oplossing niet op alle punten even diep uitgewerkt. Voordat tot realisatie kan worden overgegaan is een vervolgonderzoek dan ook noodzakelijk. Uit de validatiegesprekken kan worden geconcludeerd dat de beschreven oplossing realistisch is. In het vervolgonderzoek moeten de feedbackpunten worden meegenomen.

Met dit onderzoek is een basis voor de toekomst gelegd, waarin overzicht van het testplatform, automatisering van herhalende taken, beschikbaarheid van testomgevingen en gebruiksvriendelijkheid centraal staan. Echter, alleen met het juiste vervolgtraject kan het toekomstgerichte testplatform succesvol worden gerealiseerd. De belangrijkste aanbeveling is dan ook om zo snel mogelijk te starten met het vervolgonderzoek.

Bibliografie

[1] Dirk Krafzig, Karl Banke, en Dirk Slama. *Enterprise* SOA*: Service-Oriented Architecture Best Practices*. Prentice Hall, Upper Saddle River, New Jersey, 1e druk, 2004.

[2] IBM. *WebSphere Application Server*. `http://www.ibm.com/software/webservers/appserv/was/` (Geraadpleegd op 29 maart 2012).

[3] Glassfish. *Open Source Application Server*. `http://glassfish.java.net/` (Geraadpleegd op 29 maart 2012).

[4] Dieter Wackerow. MQS*eries Primer*. Redpaper REDP-0021-00, IBM, Oktober 1999. `www.redbooks.ibm.com`.

[5] David A. Chappell. *Enterprise Service Bus*. O'Reilly Media, Inc, Sebastopol, 1e druk, 2004.

[6] Martin Pol, Ruud Teunnisen, en Erik van Veenendaal. *Testen volgens* TM*ap*. Tutein Nolthenius, 's-Hertogenbosch, 2e druk, 2000.

[7] Ken Schwaber en Mike Beedle. *Agile Software Development with Scrum*. Prentice Hall, Upper Saddle River, New Jersey, 1e druk, 2002.

[8] Nick Rozanski en Eoin Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, Upper Saddle River, New Jersey, 2e druk, 2006.

[9] Karl E. Wiegers. *Software Requirements*. Microsoft Press, Redmond, 2e druk, 2003.

[10] Philippe Kruchten. *Architectural Blueprints – The "4+1" View Model of Software Architecture*. IEEE *Software*, 12(6):42–50, 1995.

[11] Len Bass, Paul Clements, en Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, Boston, 2e druk, 2003.

[12] Paul Clements, Flelix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, en Judith Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Boston, 1e druk, 2002.

[13] Peter Mell en Timothy Grance. *The* NIST *Definition of Cloud Computing*. NIST *Special Publication*, (800-145):1–3, 2011.

[14] Zhi Xian Chen, Seiichiro Imazeki, Matthew Kelm, Simon Kofkin-Hansen, Zhi Qiang Kou, Bobby McChesney, en Carla Sadtler. IBM *Workload Deployer: Pattern-based Application and Middleware Deployments in a Private Cloud*. Redbook SG24-8011-00, IBM, Maart 2012. `www.redbooks.ibm.com`.

[15] Amazon web services. *Amazon Virtual Private Cloud*. `http://aws.amazon.com/vpc/` (Geraadpleegd op 16 mei 2012).

[16] George Reese. *Cloud Application Architectures*. O'Reilly Media, Inc, Sebastopol, 1e druk, 2009.

[17] IBM. IBM *SmartCloud Application Services*. `http://www.ibm.com/cloud-computing/us/en/paas.html` (Geraadpleegd op 16 mei 2012).

[18] ZeroTurnaround. *Survey Results: Java EE Containers - Heaven or Hell?* `http://zeroturnaround.com/labs/java-ee-container-heaven-hell-survey-results/` (Geraadpleegd op 21 mei 2012).

[19] XebiaLabs. *Deployit*. `http://www.xebialabs.com` (Geraadpleegd op 21 mei 2012).

[20] Apache Software Foundation. *Apache Ant*. `http://ant.apache.org` (Geraadpleegd op 12 juni 2012).

[21] Twitter. *Bootstrap*. `http://twitter.github.com/bootstrap/` (Geraadpleegd op 20 april 2012).

[22] Jan van Bon, Mike Pieper, Annelies van der Veen, en Tieneke Verheijen. *Foundations of IT Service Management: based on* ITIL. Van Haren Publishing, Zaltbommel, 2e druk, 2007.

[23] Jan van Bon, Arjen de Jong, Axel Kolthof, Mike Pieper, Ruby Tjassing, Annelies van der Veen, en Tieneke Verheijen. IT *Service Management Based on* ITIL *V3 – A Pocket Guide*. Van Haren Publishing, Zaltbommel, 1e druk, 2008.

[24] Algemene Rekenkamer. *Lessen uit* ICT*-projecten bij de overheid; Deel B*. 2008. `http://www.rekenkamer.nl/Publicaties/Onderzoeksrapporten`.

[25] J.P.H. Donner. *Kamerbrief over cloud computing*, April 2011. `http://www.rijksoverheid.nl/documenten-en-publicaties/kamerstukken/2011/04/20/kamerbrief-over-cloud-computing.html`.

Afkortingen en begrippen

8-Straat

Een parallelle straat die altijd een productieversie bevat, dit om te zorgen dat van elke applicatie een productieversie beschikbaar is.

9-Straat

Een parallelle straat die wordt gebruikt voor het snel doorvoeren van een bugfix. Dit wordt ook wel de bugfixstraat genoemd.

Actiewoord

Wordt gebruikt binnen het DUO-testdossier en is een opdracht om een aantal onderling samenhangende testhandelingen uit te voeren. Een actiewoord is gekoppeld aan een specifieke reeks uit te voeren testhandelingen. De testdata waarmee deze handelingen worden uitgevoerd zijn variabel en worden vastgelegd in de bijbehorende argumenten.

Afgeleide omgeving

Om meerdere testers gelijktijdig te laten testen zijn er per applicatie per omgeving meerdere kopieën beschikbaar; bij DUO bekend als afgeleide omgevingen. De programmatuur en databasestructuur zijn in de afgeleide omgevingen allemaal gelijk aan de versie van de hoofdomgeving. Elke afgeleide omgeving heeft echter wel zijn eigen database met eigen inhoud.

AIC – *Architectuur, Integratie en Configuratiebeheer*

Vakgroep bij het Softwarehuis die de gewenste ontwikkelrichtingen aangeeft, technische kennis heeft over de communicatie tussen platformen en de kern van het SCC vormt.

↳ Zie ook: **SCC, Softwarehuis**.

Applicatieserver

Een server die software voorziet van diensten zoals beveiliging, data-services, transactie-ondersteuning, load balancing, en het beheer van grote gedistribueerde systemen. DUO maakt gebruik van GlassFish en WebSphere 6, maar zal uiteindelijk overstappen op enkel WebSphere 8.

AS/400

Een midrangesysteem van IBM dat bij DUO wordt gebruikt voor de DB2-databases en een aantal legacyapplicaties. Dit platform wordt beheerd door PROBE.

↳ Zie ook: **DB2-database, PROBE**.

Bevinding

Een geconstateerd verschil tussen een verwachting of voorspelling en de feitelijke uitkomst. Bij een functionele test is dit het verschil tussen het functioneel ontwerp en de programmatuur.

Checklist bouwtest

Deze checklist bevat de punten die door de ontwikkelaar getest moeten zijn en beschrijft enkele eisen waar deze tests aan moet voldoen.

CI – *Configuration Item*

Een component dat deel uitmaakt van of direct gerelateerd is aan de IT-infrastructuur.

CMDB – *Configuration Management Database*

Een gegevensverzameling waarin informatie met betrekking tot CI's wordt vastgelegd en beheerd.

↳ Zie ook: **CI**.

CVS – *Concurrent Versions System*

Een versiebeheersysteem voor een verzameling (bron)bestanden. Op een enkele plaats, de repository, worden alle versies van alle bestanden bijgehouden, terwijl gebruikers op een lokale kopie daarvan werken.

DB2-database

Een relationele database ontwikkeld door IBM. Het product is beschikbaar voor een groot aantal besturingssystemen en wordt bij DUO gebruikt op het AS/400 platform.

↳ Zie ook: **AS/400**.

DUO – *Dienst Uitvoering Onderwijs*

Uitvoeringsorganisatie van de Rijksoverheid voor het onderwijs. De organisatie heeft een formatieomvang van circa 1800 FTE.

↳ Zie ook: **FTE**.

ESB – *Enterprise Service Bus*

De integratie-infrastructuur (middleware) die nodig is om een SOA te faciliteren. Een ESB standaardiseert de wijze waarop service-aanvragers communiceren met service-aanbieders.

↳ Zie ook: **SOA**.

EXP – *Exploitatie*

Een testomgeving die door de afdeling I&E gebruikt wordt om te testen of de software gereed is om in productie te worden genomen.

↳ Zie ook: **I&E, Testomgeving**.

FAT – *Functionele test*

Naam voor de omgevingen die door de functioneel tester wordt gebruikt voor het uitvoeren van de functionele test.

↳ Zie ook: **FT, Testomgeving**.

FT – *Functionele test*

Een test die wordt uitgevoerd door een functioneel tester in de FAT-omgeving. Tijdens deze test wordt bepaald of het systeem werkt conform de functionele specificaties.

↳ Zie ook: **FAT**.

FTE – *Fulltime-Equivalent*

Rekeneenheid waarmee de omvang van een dienstverband of de personeelssterkte wordt uitgedrukt.

GAT – *Gebruikersacceptatietest*

1. Naam voor de testomgeving waarin de gebruikersacceptatietest wordt uitgevoerd.
2. Een test die wordt uitgevoerd door zowel de eindgebruiker als een functioneel beheerder van het systeem. Tijdens deze test wordt bepaald of het systeem de gebruiker voldoende ondersteund.

↳ Zie ook: **Testomgeving**.

Hypervisor

Wordt ook wel virtual machine manager genoemd. Dit is een hardware virtualisatietechniek die het mogelijk maakt om meerdere besturingssystemen tegelijkertijd op een host computer te laten draaien.

ICT-Demand

Afdeling bij DUO die de bedrijfsprocessen ondersteund door te voorzien in de informatiebehoefte. De afdeling is verantwoordelijk voor de acceptatie van zowel nieuwe als gewijzigde programmatuur en het geven van ondersteuning aan de gebruikers.

ICT-Regie

Afdeling bij DUO die projectleiding, SLM en contractmanagement levert en regisseert.

↳ Zie ook: **SLM**.

I&E – *Infrastructuur & Exploitatie*

Afdeling bij DUO die verantwoordelijk is voor de ontwikkeling, instandhouding en beheer van de technische infrastructuur en voor de implementatie en exploitatie van systemen op de technische infrastructuur.

IDE – *Integrated Development Environment*

Software(suite) die een ontwikkelaar ondersteunt bij het ontwikkelen van software. Het bestaat vaak uit een broncode editor, compiler en een debugger.

Keten

Een verzameling van actoren die zijn geordend volgens een logistiek proces. Deze aaneenschakeling van processen is gericht op het gezamenlijk bereiken van een resultaat. Bijvoorbeeld een service die andere service nodig heeft om een juist resultaat terug te geven.

LISB – *Legacy Integration Service Bus*

Een architectuurlaag die ervoor zorgt dat legacyapplicaties als service te benaderen zijn.

LOC – *Lines of Code*

Rekeneenheid waarmee het aantal regels broncode van een programma wordt uitgedrukt.

LSD – *Lokaal Snel Draaien*

Een door DUO ontwikkelde invoegtoepassing voor Excel die geautomatiseerd testen lokaal (vanaf de eigen PC) mogelijk maakt.

NT-Beheer

Afdeling bij DUO die het applicatief technisch beheer op het Windows platform verzorgt.

Omgeving

Een omgeving is een samenstelsel van componenten zoals hardware, software, communicatiemiddelen, procedures en de faciliteiten voor het uitvoeren van een bepaald soort taken.

OTAP – *Ontwikkeling, Test, Acceptatie en Productie*

Een software-ontwikkelmethodiek waarbij het pad van ontwikkeling, test, acceptatie en productie wordt doorlopen.

Overzetdocument

Ander woord voor vrachtbrief.

↳ Zie ook: **Vrachtbrief**.

PRD – *Productie*

De omgeving waarin zich de definitieve en geaccepteerde versies van de software bevinden. Er worden hier geen tests uitgevoerd. Dit zijn de versies die door de eindgebruikers worden gebruikt.

Private cloud

Een cloudinfrastructuur voor een enkele organisatie. Voor het beheer en de hosting van de cloud kan zowel worden gekozen voor een externe partij als het intern hosten in een eigen datacenter.

Productbacklog

Een Scrum-term die gebruikt wordt voor een op prioriteit gesorteerde lijst met functionele eisen. De lijst wordt beheerd door de product owner, die op deze manier prioriteiten kan veranderen en functionaliteiten kan toevoegen.

↳ Zie ook: **Product owner, Scrum**.

Product owner

Een rol binnen de Scrum ontwikkelmethodiek. De product owner representeert de wensen van de stakeholders door het beheren van de productbacklog. Tevens is de product owner aanwezig bij de demo van iedere sprint om het opgeleverde resultaat te accepteren.

↳ Zie ook: **Productbacklog, Scrum**.

PT – *Proeftuin*

Een testomgeving met een productieversie van de programmatuur. Deze is bedoelt voor externe partijen die op deze manier hun communicatie met de systemen van DUO kunnen testen.

RFC – *Request For Change*

Een document dat een aanvraag bevat voor een aanpassing van of aan een systeem. Het wordt in de huidige situatie ook gebruikt voor het een aanvragen van omgevingen.

RUP – *Rational Unified Process*

Een ontwikkelmethodiek voor een iteratieve en incrementele ontwikkeling van software. Het is gebaseerd op het analyseren van risico's en het verkleinen van deze risico's. Een RUP-proces bestaat uit het uitvoeren van cycli, waarbij elke cyclus eindigt met de oplevering van een nieuwe versie van het product.

Scrum

Een Agile ontwikkelmethodiek dat zich primair richt op de business value. Er wordt gewerkt in multidisciplinaire teams die in korte sprints (iteraties van twee tot vier weken) werkende software opleveren. Na iedere sprint kunnen de wensen en prioriteiten vanuit de business voor verdere ontwikkeling worden bijgesteld. Zie ook bijlage C.

↳ Zie ook: **Product owner, Scrum master**.

Scrum master

Een rol binnen de Scrum ontwikkelmethodiek die verantwoordelijk is voor het succesvol toepassen van Scrum.

↳ Zie ook: **Scrum**.

SLA – *Service Level Agreement*

Een overeenkomst tussen de klant en de IT-dienstverlener waarin overeengekomen kwaliteitsniveaus van de dienstverlening gedetailleerd en meetbaar zijn vastgelegd. Dit is beschreven met niet-technische termen, zodat het voor de beide partijen duidelijk is wat er kan worden verwacht.

SLM – *Service Level Management*

Het proces om de kwaliteit te beheersen van geleverde IT-diensten op basis van een schriftelijke overeenkomst tussen een IT-dienstverlener en haar afnemers.

SOA – *Service Oriented Architecture*

Een set van principes en methoden voor het ontwerpen en ontwikkelen van software in de vorm van interoperabele diensten.

Softwarehuis

Afdeling bij DUO die software analyseert, ontwerpt, ontwikkelt en test.

SSO – *Single sign-on*

Een techniek die gebruikers in staat stelt om eenmalig in te loggen waarna automatisch toegang wordt verschaft tot andere applicaties en resources in het netwerk.

Testdossier

Een door DUO opgesteld Excel-document dat gebruikt wordt voor het vastleggen van testgevallen en de resultaten hiervan.

Testgevallenbeschrijving GAT

Een sjabloon in Excel met een beschrijving bestaande uit algemene informatie (proces, onderdeel, versie, etc.), de testgevallenbeschrijving in logische zin, de fysieke testvulling (database-inhoud) en het testscript (draaiboek, handelingen, etc.). In het sjabloon is specifieke gebruiksinformatie als toelichting opgenomen.

Testomgeving

Een testomgeving is de combinatie van hardware (zoals een server of virtuele server) en software (zoals het besturingssysteem, frameworks, libraries en databases) waarop een test kan worden uitgevoerd.

Testplatform

Een testplatform omvat de testomgevingen en de ondersteunende architectuur, tooling en beheersaspecten om het testproces te faciliteren. Het vormt de schakeling tussen de ontwikkeling van software en het in productie nemen van deze software.

TMAP – *Test Management Approach*

Een testaanpak ontwikkeld door Sogeti voor het gestructureerd testen van software.

TPB – *Testplatform Beheer*

De werktitel die in dit onderzoek wordt gebruikt voor de afdeling die verantwoordelijk moet zijn voor het beheer van het testplatform.

TPM – *Testplatform Manager*

De werktitel die in dit onderzoek wordt gebruikt voor de maatwerksoftware van het toekomstgerichte testplatform.

Vrachtbrief

Excel-document dat wordt gebruikt voor overzettingen tussen twee opeenvolgende omgevingen in de ontwikkelstraat. In dit document is alle informatie benodigd voor de overzetting beschreven. De vrachtbrief wordt ook wel overzetdocument genoemd.

VT – *Veldtest*

Een afgeleide GAT-omgeving die gebruikt wordt om een versie van de software met externe partijen te testen.

WAS – *WebSphere Application Server*

Een applicatieserver ontwikkeld door IBM.

↳ Zie ook: **Applicatieserver**.

WSDL – *Web Service Definition Language*

Een op XML gebaseerde taal voor het beschrijven van de interfaces voor webservices.

WSF – *Wet Studiefinanciering*

Een door DUO ontwikkeld softwaresysteem dat gebruik wordt voor het uitvoeren van de wet studiefinanciering. De software is geschreven in de programmeertaal COBOL en telt ongeveer 1,5 miljoen LOC. Daarmee is dit het grootste legacysysteem dat DUO nog in gebruik heeft.

↳ Zie ook: **LOC**.

Lijst van figuren

Lijst van tabellen

A Project Initiatie Document

rijksuniversiteit groningen
faculteit wiskunde en natuurwetenschappen

Een toekomstgericht testplatform
voor Dienst Uitvoering Onderwijs

Project Initiatie Document

door

Johan van der Geest en Mark Ettema

Versie 1.0 — 23 maart 2012

Dienst Uitvoering Onderwijs
Ministerie van Onderwijs, Cultuur en Wetenschap

Documenthistorie

Revisies

Versie	Datum	Status	Wijzigingen
0.1	13-02-2012	Concept	Eerste versie, documentstructuur, introductie
0.2	14-02-2012	Concept	Doelstellingen, aanpak, organisatiestructuur, product decompositie, risicologboek
0.3	15-02-2012	Concept	Projectplan, communicatieplan, faseoverzicht
0.4	17-02-2012	Concept	Businesscase, eindproducten, randvoorwaarden, afhankelijkheden
0.5	20-02-2012	Concept	Achtergrond, projectbeheer
0.6	22-02-2012	Concept	Laatste (lay-out)aanpassingen
0.7	24-02-2012	Concept	Afronding projectplanning en kwaliteitsplan
0.8	01-03-2012	Concept	Verbeterpunten van Arjen Wassink verwerkt
0.9	09-03-2012	Concept	Aanpak en planning gewijzigd, risico toegevoegd
0.10	12-03-2012	Concept	Verbeterpunten Willem Hobers verwerkt
0.11	20-03-2012	Concept	Verbeterpunten Alexandru Telea verwerkt
1.0	23-03-2012	Final	Laatste kleine aanpassingen voor de final versie

Goedkeuringen

Versie	Datum	Naam	Functie	Handtekening
1.0	10-04-2012	Alexandru Telea	Afstudeerbegeleider RUG	
		Arjen Wassink	Afstudeerbegeleider DUO	
		Willem Hobers	Afstudeerbegeleider DUO	

Distributie

Versie	Datum	Naam	Functie
0.7	24-02-2012	Arjen Wassink	Afstudeerbegeleider DUO
		Willem Hobers	Afstudeerbegeleider DUO
0.8	01-03-2012	Arjen Wassink	Afstudeerbegeleider DUO
		Robbert Jan van Meenen	Geïnteresseerde DUO
		Willem Hobers	Afstudeerbegeleider DUO

Versie	Datum	Naam	Functie
0.9	09-03-2012	Willem Hobers	Afstudeerbegeleider DUO
		Arjen Wassink	Afstudeerbegeleider DUO
0.10	12-03-2012	Alexandru Telea	Afstudeerbegeleider RUG
		Arjen Wassink	Afstudeerbegeleider DUO
		Willem Hobers	Afstudeerbegeleider DUO
0.11	20-03-2012	Alexandru Telea	Afstudeerbegeleider RUG
		Andries Meskens	Geïnteresseerde DUO
		Arjen Wassink	Afstudeerbegeleider DUO
		Marko Ketellapper	Geïnteresseerde DUO
		Willem Hobers	Afstudeerbegeleider DUO
1.0	23-03-2012	Alexandru Telea	Afstudeerbegeleider RUG
		Arjen Wassink	Afstudeerbegeleider DUO
		Willem Hobers	Afstudeerbegeleider DUO

Inhoudsopgave

1 Inleiding

In de inleiding staat in het kort het doel en de opbouw van het PID (Project Initiatie Document) beschreven.

1.1 Doel van het document

Het PID bevat alle informatie die de opdrachtgever en afstudeerbegeleiders nodig hebben om de uitvoering van het project te kunnen autoriseren. Het is de basis voor het beoordelen van het succes en is te beschouwen als een contract tussen de opdrachtgever en uitvoerders. Het PID verankert de volgende zaken:

- het leggen van een goede basis voor het managen van het project en het beoordelen van het succes
- het verschaffen van eenduidigheid
- het voortdurend beoordelen van voortgang, problemen en levensvatbaarheid van het project.

Het PID geeft antwoord op een aantal cruciale vragen van het project:

- wat wil men met het project bereiken?
- waarom is het belangrijk om dit te bereiken?
- wie zijn er bij betrokken en wat zijn de verantwoordelijkheden?
- hoe en wanneer gaat het een en ander gebeuren?

1.2 Opbouw van het document

Het PID is opgesplitst in twee delen: een statisch en dynamisch deel. Het dynamische deel zal gedurende het project gewijzigd kunnen worden.

Het “statische” deel bevat de volgende hoofdstukken en bijlagen:

- Achtergrond (hoofdstuk 2)
- Projectdefinitie (hoofdstuk 3)
- Organisatiestructuur (hoofdstuk 4)
- Projectbeheer (hoofdstuk 5)
- Communicatieplan (bijlage A)
- Kwaliteitsplan (bijlage B)

Het “dynamische” deel bevat de volgende bijlagen:

- Businesscase (bijlage C)
- Projectplan (bijlage D)
- Risicologboek (bijlage E)

2 Achtergrond

DUO (Dienst Uitvoering Onderwijs) is de uitvoeringsorganisatie van de Rijksoverheid voor het onderwijs. De missie van DUO is het financieren en informeren van onderwijsdeelnemers en onderwijsinstellingen en het organiseren van examens.

De organisatie is gevestigd in Groningen en Zoetermeer. In Groningen bevindt zich het hoofdkantoor en nog een aantal andere vestigingen. DUO heeft een formatieomvang van circa 1800 FTE (Fulltime-Equivalent). Jaarlijks worden ongeveer 300.000 ICT gerelateerde projecturen gemaakt. De volgende afdelingen zijn voor dit project van belang:

- **ICT-Regie** (60 FTE) stuurt projecten en uitbestedingen aan. Het verzorgt de projectleiding en SLM (Service Level Management) van ICT-projecten.
- **I&E** (**Infrastructuur & Exploitatie**, 120 FTE) is verantwoordelijk voor de ontwikkeling en beheer van de technische infrastructuur. Ook verzorgt het de implementatie van systemen op deze infrastructuur.
- Het **Softwarehuis** (200 FTE) analyseert, ontwerpt, ontwikkelt en test op maat gemaakte software voor DUO. Het Softwarehuis bevindt zich op een andere locatie in Groningen dan het hoofdkantoor.
- **ICT-Demand** (220 FTE) zorgt voor de koppeling tussen ICT en de business. De afdeling regisseert de ICT en zorgt ervoor dat applicaties daadwerkelijke waarde toevoegingen leveren aan de eindgebruikers.

Het grootste softwaresysteem dat DUO in gebruik heeft, WSF (Wet Studiefinanciering), stamt uit 1986. Het is geschreven in de programmeertaal COBOL en telt ongeveer 1,5 miljoen LOC (Lines of Code). Naast COBOL heeft DUO ook een aantal systemen in beheer die geschreven zijn in COOL:2E, COOL:Plex en Java.

De toekomstvisie van DUO is om legacy applicaties te vervangen door opgesplitste diensten ontwikkeld in Java. Deze diensten bevinden zich in een SOA (Service Oriented Architecture) en moeten beschikbaar worden gesteld via een ESB (Enterprise Service Bus). Daarnaast moeten projecten meer gebruik maken van Scrum in plaats van de watervalmethodiek.

Definitie van Enterprise Service Bus

Om services aan te kunnen roepen is een voorziening nodig die de communicatie tussen serviceaanroeper (consumer) en serviceaanbieder (provider) verzorgt. Een ESB is een verzameling infrastructurele voorzieningen die berichten afhandelen, routeren en transformeren.

3 Projectdefinitie

In het vorige hoofdstuk is de achtergrond van het project beschreven. Dit hoofdstuk bevat meer details en geeft antwoord op een aantal belangrijke vragen die nodig zijn om de basis te leggen voor een controleerbaar project. De doelstellingen, aanpak, project scope, eindproducten, afhankelijkheden, randvoorwaarden en aannames worden besproken.

3.1 Doelstellingen

De doelstelling wordt met behulp van een aantal vraagstellingen beantwoord.

Waarom dient dit project te worden gerealiseerd?

Bij DUO lopen veel projecten waarin software ontwikkeld wordt vertraging op omdat een testomgeving niet op orde is.

Definitie van Testomgeving

Een testomgeving is een samenstelsel van componenten zoals hardware, software, communicatiemiddelen, procedures en de faciliteiten voor opbouw en gebruik van bestanden waarin een test wordt uitgevoerd.

DUO herkent de volgende problemen op het gebied van testomgevingen:

- **beschikbaarheid**: het niet tijdig beschikbaar zijn van een juiste, werkende testomgeving
- **traceerbaarheid**: door de beweging van grote applicaties naar opgesplitste services in een SOA is het lastiger om problemen te traceren in een keten van services
- **omgevingbeheer**: er is geen integraal beeld van wie wanneer gebruik wil maken van een testomgeving
- **versiebeheer**: er is geen overzicht van welke versies van de (gedeelde) componenten en (gedeelde) data in de omgevingen aanwezig zijn
- **verantwoordelijkheid**: er is geen duidelijkheid over wie verantwoordelijk gemaakt moet worden voor het beheer van de testomgevingen.

Hoe ziet de gewenste situatie er uit?

De gewenste situatie is een testplatform:

- waarbij terugkerende handmatige acties zoveel mogelijk vervangen zijn door automatische oplossingen
- die meer inzicht geeft in de testresultaten en eventuele problemen die optreden tijdens het testen van software
- die beter aansluit op de nieuwe technieken en methodieken die binnen DUO worden gebruikt (Java, SOA, ESB en Scrum)
- met een duidelijke eigenaar, afspraken en verantwoordelijkheden.

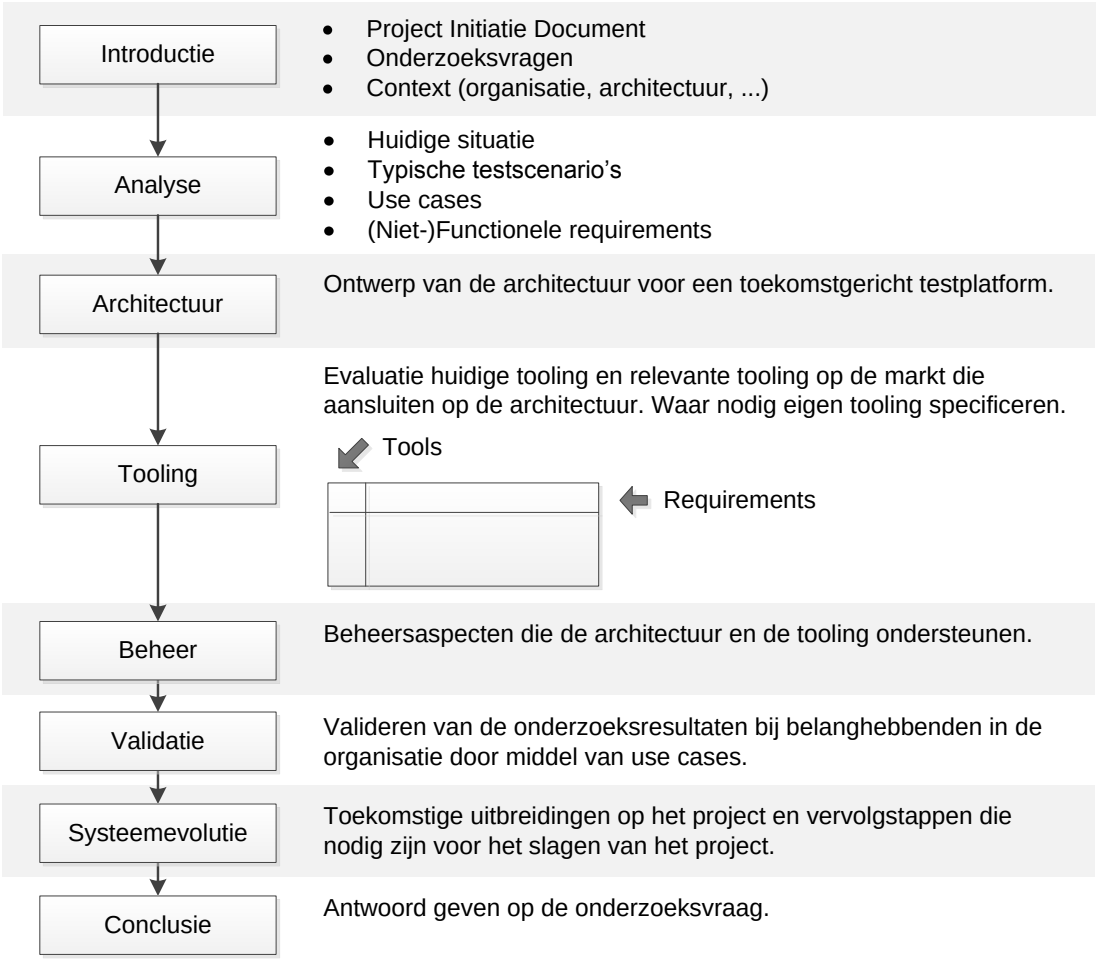
Welke voordelen brengt dit project met zich mee?

Dit project brengt de volgende voordelen met zich mee:

- **kostenefficiëntie**: er zal op vrijwel elk softwareproject tijd en dus ook kosten worden bespaard op het gebied van testen
- **reactiesnelheid**: DUO zal sneller wijzigingen in software kunnen doorvoeren die door opdrachtgevers (zoals het ministerie van OCW) worden opgelegd
- **software-evolutie**: DUO zal beter voorbereid zijn op de toekomst door een testplatform onderzocht te hebben die zich richt op Java, SOA, ESB en Scrum.

3.2 Aanpak

De aanpak van het project staat in figuur 3.1 afgebeeld. In het projectplan (bijlage D) worden de fasen van het project in meer detail besproken.



Figuur 3.1: Aanpak van het project. Aan de zijkant staat per onderdeel een korte toelichting of illustratie vermeld.

3.3 Project scope

Gezien de omvang van de software die door DUO ontwikkeld en getest wordt en de korte tijd die er voor dit project beschikbaar is, is het van belang dat er wordt afgebakend. DUO heeft een duidelijke toekomstvisie voor de ontwikkeling van software. Omdat dit project zich focust op een toekomstgericht testplatform voor DUO, is het belangrijk om het testplatform af te stemmen op deze visie.

Het project zal zich afbakenen tot een ICT- en testomgeving waarin:

- ontwikkeld wordt met de programmeertaal Java
- een SOA centraal staat als architectuur
- services worden aangeboden via een ESB
- gebruik wordt gemaakt van de softwareontwikkelmethodiek Scrum.

De volgende onderdelen zullen *geen* onderdeel vormen van het project:

- de implementatie van het testplatform
- ondersteuning voor software die ontwikkeld is met andere talen dan Java (bijvoorbeeld COBOL, COOL:2E en COOL:Plex); ofwel *legacy* software
- het onderzoeken en toetsen van TMap (Test Management Approach); een methodiek dat door DUO wordt gebruikt voor het testen van software
- een uitgebreide kosten-batenanalyse.

3.4 Eindproducten

Het project zal een afstudeerscriptie, eindpresentatie en leerpuntenrapport opleveren. Wanneer blijkt dat de onderzochte tools niet voldoende requirements dekken en er custom tooling nodig is, zal een specificatie hiervan worden opgeleverd. In het projectplan (bijlage D) staan de producten in meer detail beschreven.

3.5 Afhankelijkheden

Het project heeft de volgende afhankelijkheden:

- de bereidheid van DUO medewerkers (denk aan softwareontwikkelaars, testers en procesbeheerders) om hun mening over de huidige testomgevingen en wensen voor een toekomstig testplatform te delen
- de beschikbaarheid van documentatie over de huidige architectuur, infrastructuur en testomgevingen van DUO
- de beschikbaarheid van probeerversies van commerciële tooling.

3.6 Randvoorwaarden

Het project heeft de volgende randvoorwaarden:

- het project zal op 6 juli 2012 afgerond moeten zijn
- er zijn twee werkplekken met computers beschikbaar waarop de afstudeerscriptie en andere eindproducten gemaakt kunnen worden (in het bijzonder de mogelijkheid om zelf software te kunnen installeren)

- de begeleiders vanuit DUO en RUG (zie hoofdstuk 4) zullen tijdig beschikbaar zijn wanneer dit voor het project nodig is (bij voorkeur een uiterlijke reactietijd van 7-14 dagen)
- de opdrachtgever geeft goedkeuring voor het PID en zal waar nodig belangrijke beslissingen moeten nemen
- DUO zal de beschikking geven tot literatuur voor het onderzoek (door middel van aanschaffen, lenen van een collega of lenen uit een bibliotheek).

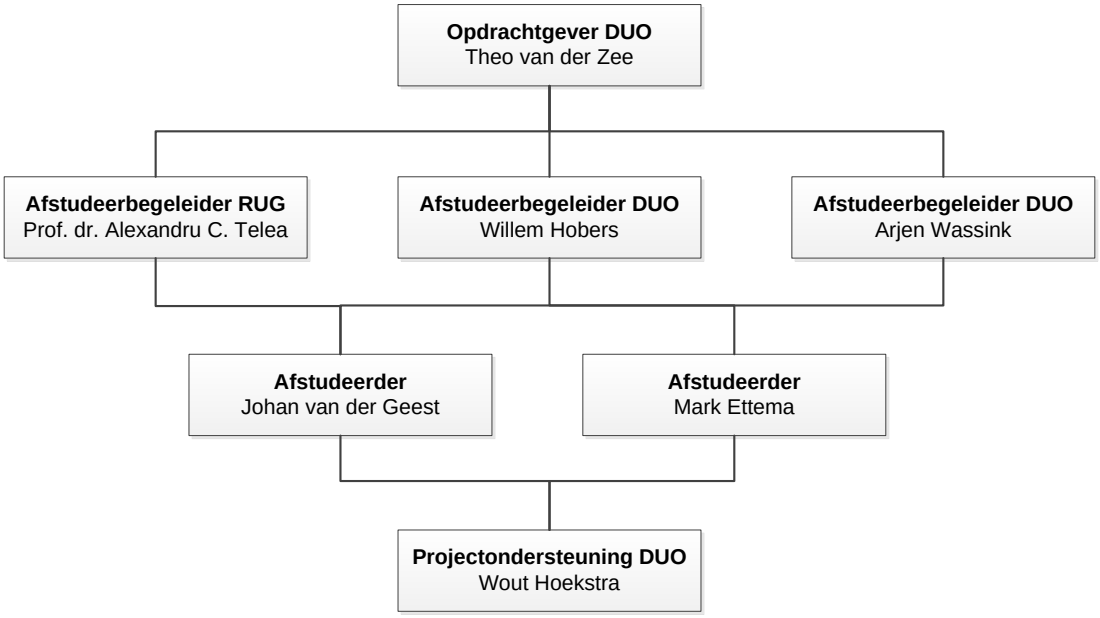
3.7 Aannames

Het project heeft de volgende aannames:

- DUO zal geen wijzigingen aanbrengen in de toekomstvisie op het gebied van ICT waarin het gebruik van Java, een SOA en een ESB centraal staat
- er zullen geen invloedrijke wijzigingen plaatsvinden op het gebied van testen, softwareontwikkeling en architectuur tussen het moment waarop de analyse is uitgevoerd en het overhandigen van de afstudeerscriptie.

4 Organisatiestructuur

De organisatiestructuur van het project staat in figuur 4.1 afgebeeld. Gedetailleerde omschrijvingen van de verantwoordelijkheden staan in de komende paragrafen beschreven.



Figuur 4.1: De organisatiestructuur van het project. De namen en functies van de personen binnen het project staan hierin aangegeven.

4.1 Opdrachtgever DUO

De opdrachtgever van dit project is Theo van der Zee. Zijn functie binnen DUO is Manager ICT-Demand. Belangrijke beslissingen voor het project zullen door hem genomen moeten worden.

4.2 Afstudeerbegeleiders DUO

Vanuit DUO zijn er twee afstudeerbegeleiders beschikbaar, namelijk Willem Hobers en Arjen Wassink. Beide personen zijn werkzaam bij DUO als Specialist Functioneel Beheer. Ze zullen sturing en ondersteuning geven waar dat nodig is. Denk hierbij aan het vinden van de juiste contactpersonen binnen DUO en andere praktische en inhoudelijke zaken omtrent het afstuderen en de opdracht.

4.3 Afstudeerbegeleider RUG

De afstudeerbegeleider vanuit de Rijksuniversiteit Groningen is prof. dr. Alexandru C. Telea. Hij zal terugkoppeling geven over de inhoud en opbouw van de afstudeerscriptie. Alexandru Telea heeft erg veel kennis over het onderwerp Software Quality Assurance & Testing en kan de afstudeerders in de juiste richting sturen als dat nodig is. Zijn rol is ook om te toetsen of de afstudeerscriptie voldoet aan de eisen van een MSc-scriptie.

4.4 Afstudeerders

Johan van der Geest en Mark Ettema studeren af bij de Rijksuniversiteit Groningen (Master in Computing Science) en voeren het project uit. Zij zijn verantwoordelijk voor het op tijd leveren van een eindproduct waar zowel DUO, de RUG als zijzelf tevreden mee zijn.

4.5 Projectondersteuning DUO

Vanuit DUO is er ook projectondersteuning beschikbaar. De projectondersteuning, geleverd door Wout Hoekstra, is enkel bedoeld voor praktische zaken, dus niet voor inhoudelijke begeleiding van het project. Denk hierbij aan:

- het regelen van specifieke software op de werkplekken
- het regelen van beveiligingstokens om op afstand te kunnen werken
- ondersteuning geven bij vervoer tussen de verschillende locaties van DUO.

5 Projectbeheer

Dit hoofdstuk beschrijft de verantwoordelijkheden, toleranties en procedures van alle partijen die betrokken zijn bij dit project.

5.1 Rapporteren

Tabel 5.1 geeft de relatie weer tussen de betrokken partijen en de projectdocumenten. Zie ook het communicatieplan (bijlage A).

	Opdracht-gever	**Afstudeer-begeleiders DUO**	**Afstudeer-begeleider RUG**	**Afstu-deerders**	**Project-onder-steuning**
PID	C + I	C + A	C + A	S + V	I
Conceptversies eindrapport	C + I	C + R	C + R	S + V	I
Eindrapport	C + I	C + R + A	R + A	S + V	I
Eindpresentatie	C + I	C + I	R + A	S + V	I
Uitzonderingsplan	C + I	C + A	C + A	S + V	I

Legenda			
S	Schrijven	**R**	Reviewen
C	Raadplegen	**V**	Verspreiden / Archiveren
A	Accepteren	**I**	Ter informatie

Tabel 5.1: Relaties tussen de betrokken partijen en de projectdocumenten. In de legenda staan de letters uit het rapportageoverzicht uitgelegd.

5.2 Risicomanagement

Risico's zullen zo snel mogelijk moeten worden besproken met de begeleiders van DUO (Willem Hobers en Arjen Wassink). Als een risico niet kan worden vermeden zal de begeleider van de RUG (Alexandru Telea) hier over worden ingelicht. Om risico's te voorkomen is een risicologboek (bijlage E) opgesteld. Dit is een lijst met mogelijke risico's en maatregelen om deze te voorkomen.

5.3 Voortgangscontrole

Tabel 5.2 geeft een overzicht met welke frequentie er voortgangsgesprekken plaatsvinden en wie daarbij aanwezig zullen zijn.

Partij	Aanwezig	Frequentie	Tijd	Doel	Onderwerpen
DUO	Johan van der Geest Mark Ettema Willem Hobers	1x per week	Variabel	Projectoverleg	Voortgang, problemen, risico's
DUO	Johan van der Geest Mark Ettema Arjen Wassink	1x per week	Variabel	Projectoverleg	Voortgang, problemen, risico's
RUG	Johan van der Geest Mark Ettema Alexandru Telea	Variabel	Variabel	Projectoverleg	Voortgang, problemen, risico's

Tabel 5.2: Een overzicht met wie, welk doel en welke frequentie er voortgangsgesprekken plaats zullen vinden.

5.4 Tolerantie

Het project moet worden afgerond voor 6 juli 2012. De Rijksuniversiteit Groningen heeft als eis dat het project volledig is afgerond binnen een jaar na het starten van het project. Hierdoor is een uitloop van maximaal zes maanden mogelijk, al zal zoveel mogelijk moeten worden voorkomen dat hier gebruik van wordt gemaakt.

Een gedetailleerd overzicht van de fasen, de planning en een beschrijving van de eindproducten is te vinden in het projectplan (bijlage D).

5.5 Afwijk- en uitzonderingsprocedures

Er moet een uitzonderingsprocedure worden gestart als blijkt dat er te veel tijd aan een bepaalde fase wordt besteed. Onderdeel van deze procedure is het opstellen van een uitzonderingsplan, waarin wordt beschreven hoe het probleem kan worden opgelost en er verder kan worden gegaan met de volgende fase. Zoals aangegeven in paragraaf 5.1 zal dit uitzonderingsplan moeten worden goedgekeurd door de begeleiders van DUO en de begeleider van de RUG.

A Communicatieplan

Het communicatieplan bevat alle belanghebbenden met een interesse voor het project en de manier waarop deze geïnformeerd zullen worden.

A.1 Belanghebbende partijen

Tabel A.1 geeft een overzicht alle belanghebbenden, hun functie en de taken die de desbetreffende persoon heeft.

Wie	Namens	Functie	Taken
Mark Ettema	Rijksuniversiteit Groningen	Afstudeerder	Adviseren, beslissen, uitvoeren
Johan van der Geest	Rijksuniversiteit Groningen	Afstudeerder	Adviseren, beslissen, uitvoeren
Alexandru Telea	Rijksuniversiteit Groningen	Afstudeerbegeleider	Adviseren, beslissen, beoordelen
Willem Hobers	Dienst Uitvoering Onderwijs	Afstudeerbegeleider	Adviseren, beslissen
Arjen Wassink	Dienst Uitvoering Onderwijs	Afstudeerbegeleider	Adviseren, beslissen
Theo van der Zee	Dienst Uitvoering Onderwijs	Opdrachtgever	Beslissen

Tabel A.1: Overzicht van de belanghebbende personen met hun bijbehorende functie en taken.

A.2 Communicatieroutes

Tabel A.2 geeft een overzicht van de soorten informatie die er tussen de verschillende partijen gedeeld zal worden en via welk medium en met welke frequentie dit zal gebeuren.

Van	Naar	Informatie	Medium	Frequentie
Johan van der Geest Mark Ettema	Alexandru Telea	Project updates	E-mail, direct	Variabel
Johan van der Geest Mark Ettema	Willem Hobers	Project updates, ondersteuning	E-mail, direct	Gemiddeld één keer per week
Johan van der Geest Mark Ettema	Arjen Wassink	Project updates, ondersteuning	E-mail, direct	Gemiddeld één keer per week
Johan van der Geest Mark Ettema	Theo van der Zee	(Tussen)resultaten	E-mail	Variabel

Tabel A.2: Overzicht van de communicatieroutes tussen de verschillende partijen en welke informatie er gedeeld zal worden.

B Kwaliteitsplan

Het doel van het kwaliteitsplan is het beschrijven van de kwaliteitseisen en standaarden waaraan het project moet voldoen. Ook wordt beschreven wie er verantwoordelijk zijn dat aan deze eisen wordt voldaan.

B.1 Acceptatiecriteria

Voor dit project zijn de volgende acceptatiecriteria vastgesteld:

- het PID is goedgekeurd door de afstudeerbegeleiders
- de eindproducten zijn opgeleverd voor 6 juli 2012
- de eindproducten zijn in het Nederlands geschreven
- de afstudeerscriptie bevat een samenvatting van maximaal één pagina
- de afstudeerscriptie is goedgekeurd door de afstudeerbegeleiders
- de onderzoeksvraag sluit aan op de probleemstelling uit de businesscase
- de conclusie geeft een duidelijk antwoord op de onderzoeksvraag
- de eindpresentatie duurt circa 45 minuten (inclusief vragen) en wordt gehouden op de RUG.

KPI's (Key Performance Indicators) voor de testomgeving kunnen in dit stadium van het project nog niet worden bepaald, omdat de requirements nog niet bekend zijn. Deze zullen tijdens de analyse worden gedefinieerd.

B.2 Verantwoordelijkheden

Het is de verantwoordelijkheid van de afstudeerders (Johan van der Geest en Mark Ettema) dat het project aan de eisen die in dit document beschreven staan zal voldoen.

C Businesscase

De businesscase is door de afdeling ICT-Demand van DUO bij de aanvang van het project aangeleverd. Op basis van de businesscase hebben de afstudeerders het project aangenomen.

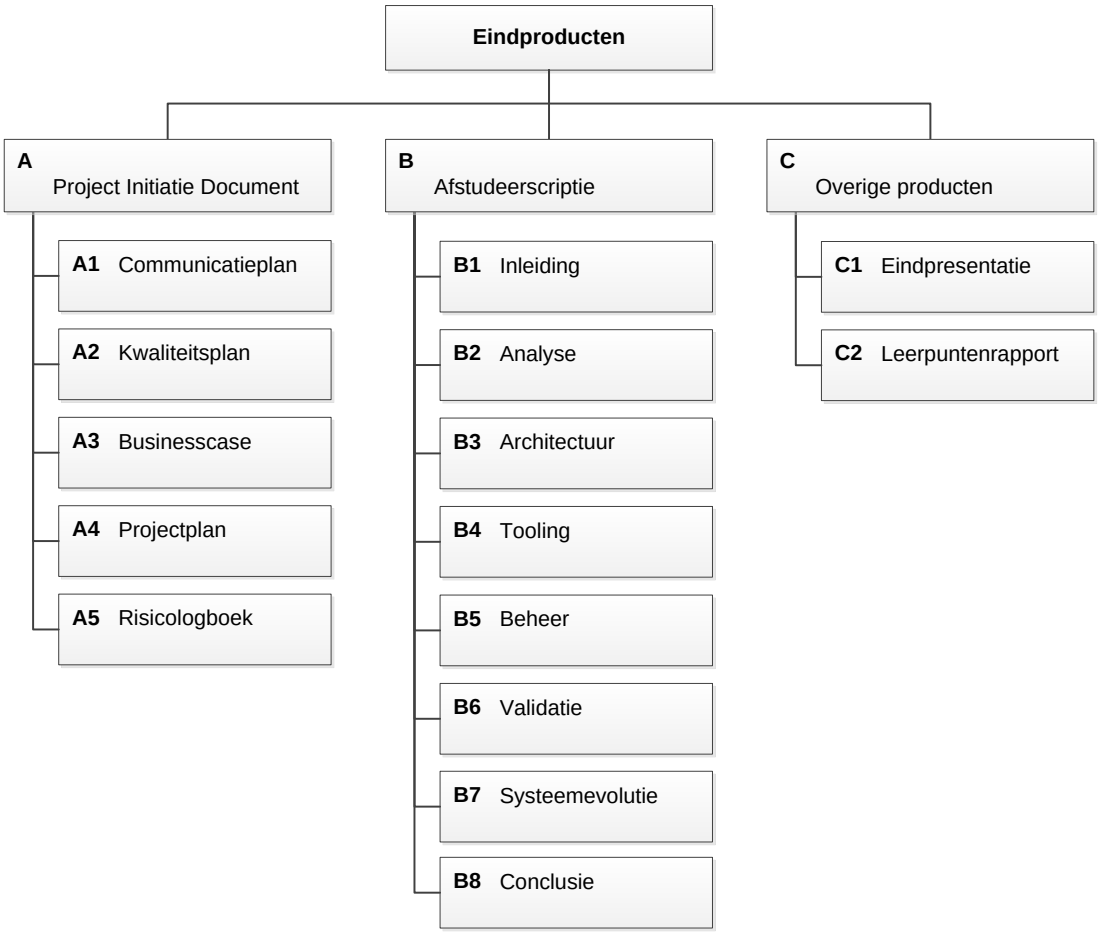
> “ Geautomatiseerde systemen vertonen steeds meer samenhang. Vernieuwingen in die systemen moeten meer en meer in samenhang getest worden. Testomgevingen zijn schaars. De inrichting van de testomgevingen moet meer en meer in samenhang gebeuren. Er is geen integraal beeld van wie wanneer gebruik wil maken van welke testomgeving, en over welke versies van welke (gedeelde) componenten en (gedeelde) data dit dient te gebeuren. De tijdige beschikbaarheid van een juiste, werkende testomgeving is op dit moment een knelpunt voor de voortgang van projecten.
>
> Wat moeten we doen om de testomgevingen optimaal en tijdig beschikbaar te hebben voor de testtrajecten van de projecten? Een andere vraag is hoe we inzichtelijk krijgen op welke wijze de verschillende testtrajecten met elkaar interfereren. Wie zou verantwoordelijk gemaakt moeten worden voor het wijzigingenbeheer op testomgevingen, wie regisseert dit omgevingenbeheer en wie voert het uit?
>
> *Vraag*: Als je opnieuw mocht beginnen hoe zou je dat dan doen, welke nieuwe structuur zou hier het beste bij passen? ”

D Projectplan

Het projectplan geeft een overzicht van de producten die gedurende het project worden opgeleverd. Dit document kan worden gebruikt om de voortgang van het project te controleren. Per fase is aangegeven wanneer er aan gewerkt zal worden. Indien nodig zal dit document gedurende het project worden bijgewerkt.

D.1 Product Decompositie Structuur

Figuur D.1 geeft de product decompositie structuur weer. Dit is een overzicht van de hiërarchie van alle producten die zullen worden opgeleverd.



Figuur D.1: De product decompositie structuur geeft een overzicht van de hiërarchie van alle producten die worden opgeleverd.

D.2 Faseoverzicht

Tabel D.1 geeft een gedetailleerde beschrijving van de verschillende fasen gedurende het project. Per fase is een omschrijving gegeven, staan (indien van toepassing) de bijbehorende producten vermeld en is aangeven in welke periode er aan gewerkt zal worden.

Fase	Product	Naam en omschrijving	Start	Eind *
0	-	**Kennismaking** Er zal kennis worden gemaakt met verschillende personen binnen de organisatie. Ook zal duidelijk worden waar informatie het beste verkregen kan worden.	06/02/12	24/02/12
1	A	**Project Initiatie Document (PID)** Dit document bevat alle belangrijke informatie die nodig is voor de start van het project.	08/02/12	16/03/12
	A1	**Communicatieplan** Het communicatieplan geeft een overzicht van de belanghebbenden, hun taken en op welke manier er gecommuniceerd wordt.	–	–
	A2	**Kwaliteitsplan** Het kwaliteitsplan beschrijft de acceptatiecriteria en wie verantwoordelijk zijn om deze kwaliteit te waarborgen.	–	–
	A3	**Businesscase** De businesscase beschrijft de redenen voor het starten van dit project.	–	–
	A4	**Projectplan** Het projectplan geeft een overzicht van de producten die worden opgeleverd en wanneer er aan gewerkt zal worden.	–	–
	A5	**Risicologboek** Het risicologboek geeft een overzicht van de risico's van het project en welke maatregelen er getroffen kunnen worden om deze risico's te voorkomen.	–	–
2	B	**Afstudeerscriptie** De afstudeerscriptie is het belangrijkste document voor de eindbeoordeling van het project. Tijdens het gehele project zal er aan gewerkt worden.	27/02/12	29/06/12
	B1	**Inleiding** Bevat de probleemstelling, onderzoeksvragen en geeft een inleiding van de organisatie, betrokken afdelingen en ICT-infrastructuur en architectuur.	27/02/12	16/03/12

Fase	Product	Naam en omschrijving	Start	Eind *
	B2	**Analyse** Er zal worden geanalyseerd hoe de huidige testomgevingen (testproces, tooling, eigenaren, etc.) zijn opgebouwd. Deze analyse zal worden ondersteund met use cases en probleempunten van de huidige situatie. Er zal ook onderzoek worden gedaan naar de wensen voor een toekomstgericht testplatform. Deze wensen worden vertaald naar functionele en niet-functionele requirements.	05/03/12	20/04/12
	B3	**Architectuur** In dit hoofdstuk zal de architectuur beschreven worden voor een toekomstgericht testplatform. Deze zal een deel van de requirements dekken. Onder de architectuur valt de opbouw van het testplatform.	09/04/12	11/05/12
	B4	**Tooling** In dit hoofdstuk zullen zowel de huidige als relevante tooling op de markt worden beoordeeld. Als custom tooling nodig blijkt te zijn, kan dit middels een proof of concept worden verduidelijkt.	23/04/12	01/06/12
	B5	**Beheer** In dit hoofdstuk zullen de beheersaspecten komen te staan die de architectuur en de tooling moeten ondersteunen. Denk hierbij aan de eigenaar, onderhoudsaspecten, definities en verantwoordelijkheden.	14/05/12	08/06/12
	B6	**Validatie** In dit hoofdstuk zal worden gevalideerd in hoeverre de gestelde requirements zijn bereikt door middel van de producten B3, B4 en B5. Dit kan met behulp van use cases worden gedaan. De validatie wordt uitgevoerd bij een aantal belanghebbenden van het project. Het is een belangrijke voorwaarde om daadwerkelijk iets met die resultaten te kunnen doen.	11/06/12	24/06/12
	B7	**Systeemevolutie** In dit hoofdstuk zullen mogelijke toekomstige uitbreidingen op het testplatform worden onderzocht. Ook zullen de vervolgstappen die nodig zijn voor het slagen van het project besproken worden.	25/06/12	29/06/12
	B8	**Conclusie** In dit hoofdstuk zal de eindconclusie beschreven worden en daarmee de onderzoeksvragen beantwoorden.	25/06/12	29/06/12

Fase	Product	Naam en omschrijving	Start	Eind *
3	C	**Overige producten** De overige producten die worden opgeleverd naar aanleiding van dit project.		
	C1	**Eindpresentatie** Het project zal worden afgerond met een eindpresentatie bij de Rijksuniversiteit Groningen.	25/06/12	06/07/12
	C2	**Leerpuntenrapport** Dit document bevat de belangrijkste leerpunten die gedurende het project naar voren zijn gekomen.	06/02/12	06/07/12

Tabel D.1: Overzicht van de fasen, bijbehorende producten en wanneer er aan gewerkt zal worden.

* **Eind**: Het is mogelijk een deadline te wijzigen middels een uitzonderingsplan.

E Risicologboek

Het risicologboek (zie tabel E.1) bevat de initiële risico's van dit project. Van elk risico is de kans dat het voorkomt en de impact ervan aangegeven met laag (L), gemiddeld (G) of hoog (H). Gedurende het project kunnen nieuwe risico's aan deze tabel worden toegevoegd.

ID	Beschrijving	Kans	Impact	Maatregelen
1	Het project blijkt te groot en complex.	H	H	Zorg voor een duidelijke afbakening van het project. Indien nodig pas de afbakening aan met behulp van een uitzonderingsplan.
2	Het project wordt teveel beïnvloed door de mening of visie van belanghebbenden.	H	H	Blijf tijdens interviews en gesprekken neutraal. Verifieer de mening bij andere partijen en doe onderzoek naar de correctheid ervan.
3	Er treedt een vertraging op in de planning van het project.	G	H	Blijft zoveel mogelijk gefocust op de projectplanning. Maak een uitzonderingsplan als er zich een risico voordoet.
4	De wensen van de opdrachtgever veranderen gedurende het project.	G	H	Zorg in de beginfase dat de wensen op papier staan en dat alle belanghebbenden hier inzicht in krijgen. Maak duidelijk dat wijzigingen gedurende het project (onnodige) vertraging oplevert en daarom moet worden voorkomen.
5	Met de resultaten van het project wordt uiteindelijk niets gedaan.	G	G	Zorg dat alle stakeholders op de hoogte blijven van de onderzoeksresulaten. Geef aanbevelingen voor vervolgstappen in de afstudeerscriptie.
6	Personen die de juiste informatie kunnen verstrekken hebben weinig tijd.	G	G	Zorg dat afspraken tijdig gemaakt worden, zodat de andere partij meer ruimte heeft om een voorkeur aan te geven.

ID	Beschrijving	Kans	Impact	Maatregelen
7	Het project focust zich teveel op de business en sluit daardoor onvoldoende aan bij de Master Computing Science en de eisen vanuit de RUG.	G	G	Maak duidelijk aan de begeleiders dat technisch onderzoek een belangrijk onderdeel is. Definieer in het begin technische requirements en redeneer in dezelfde termen verderop in de scriptie.
8	Het duurt lang voordat benodigde middelen beschikbaar zijn.	G	G	Zorg dat belangrijke middelen tijdig worden aangevraagd en houdt er rekening mee dat sommige procedures tijd kosten. Zorg wanneer nodig voor een tijdelijke oplossing.
9	Een projectlid wordt ziek voor een langere periode (twee of meer weken).	L	H	Ziekte is moeilijk om tegen te gaan, maar zorg voor een gezonde werkomgeving.
10	De besluitvorming van de opdrachtgever laat lang op zich wachten.	L	H	Zorg dat er tijdig wordt aangegeven wanneer er een besluit genomen moet worden, zodat de opdrachtgever er op voorbereid is.
11	Er vindt te weinig communicatie plaats met de opdrachtgever.	L	G	Maak gebruik van het communicatieplan. Wanneer er niet wordt gereageerd op mail, probeer het telefonisch.

Tabel E.1: Het risicologboek bevat risico's voor het project, de kans en impact hiervan en maatregelen om deze te voorkomen.

B Gesprekken

Datum	Gespreksonderwerp	Persoon	Afdeling	Functie
06-02-2012	Voorlichting applicatielandschap	Arjen Wassink	ICT-Demand	Specialist functioneel beheer
07-02-2012	Kennismaking manager ICT-Demand	Theo van der Zee	ICT-Demand	Manager ICT-Demand
08-02-2012	Kennismaking Triple O project	Peter Hoven	Projectmanagement	Projectmanager
		Jeroen Smit	ICT-Regie	Projectleider
10-02-2012	Voorlichting architectuur	Frits Bouma	Softwarehuis	Specialist softwareontwikkeling
13-02-2012	Paneltest internetwizard	Rene Glade	ICT-Demand	Senior procesbeheerder
13-02-2012	Voorlichting MQ en ESB	Edde van de Vosse	Softwarehuis	Senior programmeur / Technisch ontwerper
14-02-2012	Rondleiding datacentrum Kempkensberg	Patrick Schut	I&E	Coördinator operations
14-02-2012	Kennismaking Service Level Management	Frank Schaafsma	ICT-Demand	Business service manager
20-02-2012	Toelichting op het testproces	Erik Kwast	Softwarehuis	Specialist softwareontwikkeling
21-02-2012	Testtools voor systeemtesten	Bert Elzinga	Softwarehuis	Senior functioneel tester
29-02-2012	Testomgevingenbeheer FAT	Piet de Haan	Softwarehuis	Testnavigator
		Gerd van den Berg	Softwarehuis	Testnavigator
01-03-2012	Kennismaking project belanghebbende	Robbert Jan van Meenen	Projectmanagement	Projectmanager
06-03-2012	Toelichting op het testproces (vervolg)	Erik Kwast	Softwarehuis	Specialist softwareontwikkeling
07-03-2012	Omgevingenbeheer en infrastructuur	Frank Wilbrink	I&E	Architect infrastructuur
12-03-2012	GAT-testen Java-projecten	Edwin Huijser	ICT-Demand	Procesbeheerder
		Edo Visser	ICT-Demand	Procesbeheerder
14-03-2012	Lopende projecten omgevingenbeheer	Andries Mesken	Softwarehuis	Senior programmeur / Technisch ontwerper
		Arjen Wassink	ICT-Demand	Specialist functioneel beheer
		Frank Wilbrink	I&E	Architect infrastructuur
		Jeroen Smit	ICT-Regie	Projectleider
		Marko Ketellapper	Softwarehuis	Senior functioneel tester
		Robbert Jan van Meenen	Projectmanagement	Projectmanager
		Rudi Rondhuis	ICT-Regie	Projectleider

Datum	Gespreksonderwerp	Persoon	Afdeling	Functie
15-03-2012	Introductie PROBE	Quido van Hazendonk	I&E	Beheerder infrastructuur
19-03-2012	Kennismaking PVS	Rob van der Niet	Softwarehuis	Senior functioneel tester
22-03-2012	Testomgevingen ”Keep it simple”	Andries Mesken	Softwarehuis	Senior programmeur / Technisch ontwerper
		Marko Ketellapper	Softwarehuis	Senior functioneel tester
22-03-2012	Kennismaking Softwarecontrol	Harke Wijnsma	Softwarehuis	Technisch objectbeheerder
27-03-2012	Service- en berichtenboek	Benno van Brussel	Softwarehuis	Senior ontwerper / MTHV
		Roelof Zomers	Softwarehuis	Specialist systeemontwikkeling en beheer
29-03-2012	Testomgeving voor het PVS project	Han Duisterwinkel	ICT-Regie	Testmanager PVS
13-04-2012	Kennismaking NT-Beheer	Jan Kempen	I&E	Coördinator beheer infrastructuur
		Peter Bloed	I&E	Beheerder infrastructuur
13-04-2012	Voortgangsoverleg	Robbert Jan van Meenen	Projectmanagement	Projectmanager
19-04-2012	Kennismaking afdelingsmanager ICT-Demand	Meindert Westera	ICT-Demand	Afdelingsmanager
14-05-2012	Verificatie huidige en gewenste situatie	Andries Mesken	Softwarehuis	Senior programmeur / Technisch ontwerper
		Marko Ketellapper	Softwarehuis	Senior functioneel tester
15-05-2012	Verificatie huidige en gewenste situatie	Arjen Wassink	ICT-Demand	Specialist functioneel beheer
		Willem Hobers	ICT-Demand	Specialist functioneel beheer
16-05-2012	Verificatie huidige en gewenste situatie (vervolg)	Arjen Wassink	ICT-Demand	Specialist functioneel beheer
26-06-2012	Validatiegesprek	Andries Mesken	Softwarehuis	Senior programmeur / Technisch ontwerper
		Marko Ketellapper	Softwarehuis	Senior functioneel tester
26-06-2012	Validatiegesprek	Robbert Jan van Meenen	Projectmanagement	Projectmanager
26-06-2012	Validatiegesprek	Edwin Huijser	ICT-Demand	Procesbeheerder
27-06-2012	Validatiegesprek	Jeroen Smit	ICT-Regie	Projectleider
27-06-2012	Validatiegesprek	Piet de Haan	Softwarehuis	Testnavigator
		Gerd van den Berg	Softwarehuis	Testnavigator
		Marten de Vries	Softwarehuis	Testnavigator

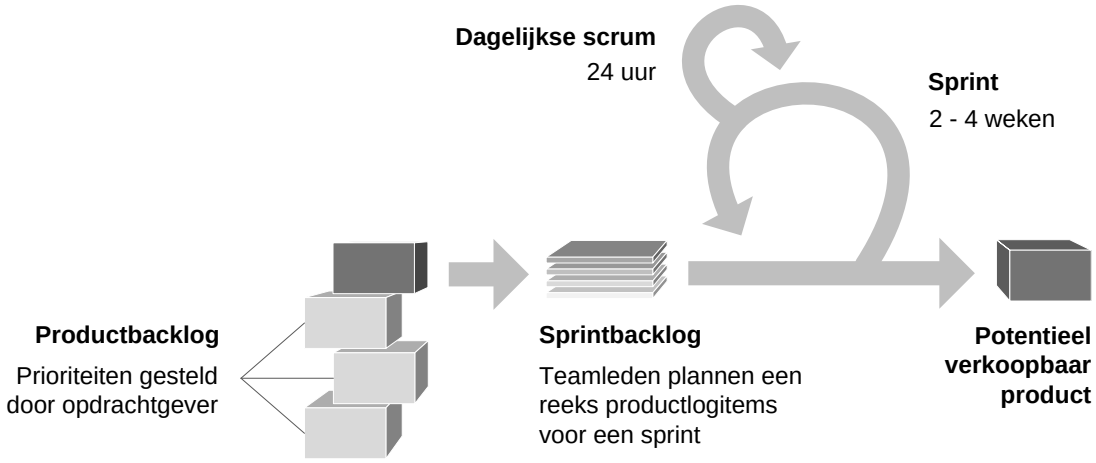
Datum	Gespreksonderwerp	Persoon	Afdeling	Functie
28-06-2012	Validatiegesprek	Harke Wijnsma	Softwarehuis	Technisch objectbeheerder
04-07-2012	Validatiegesprek	Peter Bloed	I&E	Beheerder infrastructuur

Tabel B.1: Een overzicht van de gesprekken die tijdens het afstudeeronderzoek zijn gevoerd. De lijst is gesorteerd op de datum waarop het gesprek heeft plaatsgevonden. Het bevat het gespreksonderwerp, de persoon of personen waarmee het gesprek is gevoerd en de bijbehorende afdelingen en functies.

C Scrum ontwikkelmethodiek

Scrum is een Agile ontwikkelmethodiek waarbij wordt gewerkt met sprints van twee tot vier weken. Na iedere sprint wordt een werkbare demo opgeleverd en kunnen wensen en prioriteiten voor de verdere ontwikkeling worden bijgesteld. In figuur C.1 zijn de kenmerken en de werking van Scrum weergegeven. Er zijn verschillende redenen voor het gebruik van Scrum in plaats van de watervalmethodiek:

- bij de start van een project zijn de requirements vaak nog niet volledig
- vanwege ontwikkelingen bij een organisatie kunnen de requirements gedurende het project veranderen
- een klant (opdrachtgever) weet pas precies wat hij wil als hij een eerste versie van de software heeft gezien.



Figuur C.1: De kenmerken en werking van de Scrum ontwikkelmehodiek.

Sprint voorbereiding

Door de stakeholders wordt een lijst met functionaliteiten opgesteld, de productbacklog. Deze lijst wordt beheerd door de product owner en alleen deze mag de prioriteit van de items op deze lijst bepalen. Bij de start van een nieuwe sprint wordt door het Scrumteam (dat uit vijf tot negen teamleden bestaat) in overleg met de Scrum master een planning gemaakt. Hierbij wordt bepaald hoeveel items van de productbacklog tijdens de sprint zullen worden uitgewerkt en daarmee op de sprintbacklog komen. Daarnaast wordt een sprintdoel opgesteld, zodat voor alle partijen duidelijk is wat er met de sprint bereikt gaat worden.

De sprint

Bij de start van een project wordt de exacte duur van een sprint bepaald en deze zal voor elke sprint gelijk zijn. Tijdens een sprint mag er alleen aan de items van de sprintbacklog worden gewerkt. Doordat een sprint maximaal vier weken duurt wordt geaccepteerd dat er tussentijds geen taken mogen worden toegevoegd. Tussen het Scrumteam en de Scrum master vindt dagelijks (op een vaste tijd en

locatie) een meeting van maximaal 15 minuten plaats. Hierbij worden drie vragen beantwoord:

1. Wat is er gedaan sinds de laatste meeting?
2. Wat staat er vandaag op de planning?
3. Zijn er problemen waardoor het werk wordt belemmerd?

Als er problemen zijn waardoor het werk van het Scrumteam wordt belemmerd zal de Scrum master helpen deze problemen op te lossen. Hij kan worden gezien als de coach van het Scrumteam. Om de voortgang van een sprint inzichtelijk te maken worden de taken voorzien van een status, zoals: te doen, in ontwikkeling, wordt getest en afgerond. Met behulp van een burn-down grafiek wordt het verwachte aantal uren dat nodig is om alle items van de sprintbacklog af te ronden weergegeven. Problemen of tegenvallers zijn hierdoor direct zichtbaar. Voor zowel het bijhouden van de taken als de burn-down grafiek zijn tools op de markt beschikbaar. Er wordt ook vaak met een whiteboard of post-its gewerkt, zodat de voorgang continu voor iedereen in de projectruimte zichtbaar is.

Sprint afronding

Een sprint wordt afgesloten met een demonstratie voor de product owner, stakeholders en andere geïnteresseerden. Het is belangrijk dat deze demo goed verloopt. Een goed geteste versie en een stabiele omgeving zijn daarom belangrijk. Na de demo wordt de sprint geëvalueerd. Dit wordt de sprint review genoemd. Eventuele niet afgeronde items gaan terug naar de productbacklog.

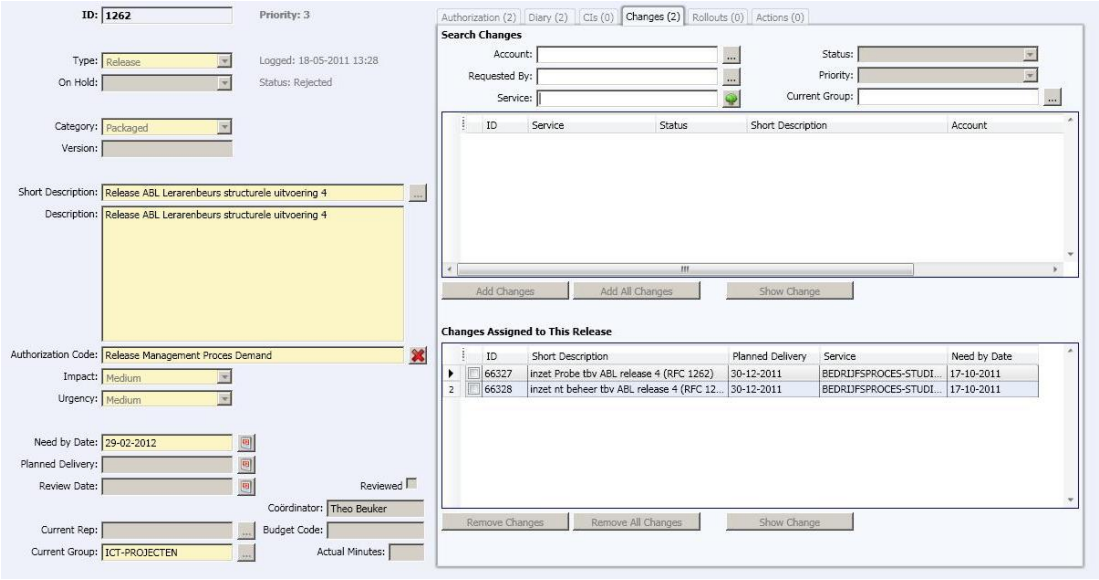
Parallel werken

Er kan met meerdere Scrumteams parallel aan een project worden gewerkt. In dat geval wordt gebruik gemaakt van dezelfde productbacklog, maar werkt ieder team aan zijn eigen geselecteerde items (sprintbacklog). Het is belangrijk dat de teams onafhankelijk van elkaar kunnen werken. Met een integratiesprint kan later de koppeling tussen de verschillende onderdelen door één van de teams worden gerealiseerd.

D Huidige tooling

D.1 ITSM

ITSM staat centraal voor het vastleggen van de administratie van de IT-dienstverlening binnen DUO. Met behulp van deze tool worden incident-, probleem- en wijzigingsprocessen ondersteund. In de CMDB worden onderlinge relaties tussen objecten vastgelegd. Figuur D.1 toont de in ITSM vastgelegde gegevens van een softwarerelease met daarbij de verwijzingen naar bijbehorende RFC's.



Figuur D.1: Een in ITSM vastgelegde softwarerelease met verwijzingen naar bijbehorende RFC's.

D.2 LSD (Lokaal Snel Draaien)

LSD is een door DUO ontwikkelde tool voor het uitvoeren van geautomatiseerde tests. Voor de bediening kan gebruik worden gemaakt van de werkbalk en de menu's die door deze invoegtoepassing aan Excel zijn toegevoegd (zie figuur D.2).

Figuur D.2: Na de installatie van de invoegtoepassing beschikt Excel over een werkbalk en menu's om LSD te kunnen bedienen.

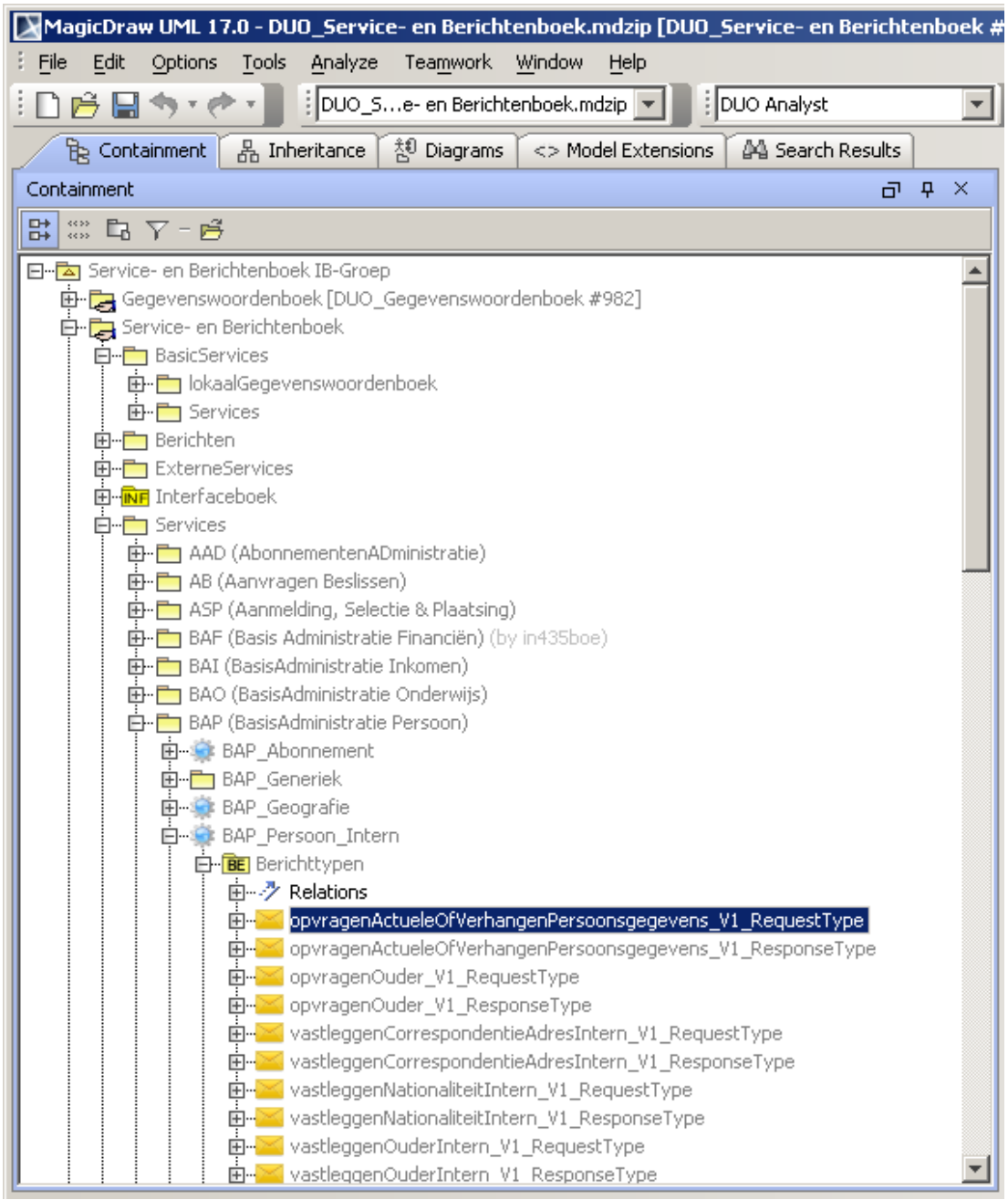
Figuur D.3 toont een voorbeeld van een met actiewoorden beschreven test die met LSD kan worden uitgevoerd.

Faw BAP UC60 Opvragen Adres			versie 3.1	analist Andries van der Velde	
* Faw BAP UC60 Opvragen Adres	*request*	*response*	*omgeving+versie*	*persoonsID*	*controle-signaalCode*
			FAT-32		
Faw BAP UC60 Opvragen Adres	3, Bestandsnaam	1, waarde	2, waarde	3, bap:persoonsID	4, Waarde
		1=bewaar waarde	naam=response		
		2=bewaar waarde	naam=omgevingversie		
		3=opvragenAdres BAP60 30 schrijf xml bestand	response=&refer[response]		
		4=controleer xml bestand	Bestand=&refer[response]	Xml tag=signaalCode	
	Start situatie:	geen			
	Eind situatie:	bericht verstuurd en gecontroleerd			
	Omschrijving:	aanroepen BAP UC60 Opvragen adres, met controle op signaal label [omgeving-versie-BAPPER] uit bap.ini			
	Wijziging:	versie	3.0		
		omschrijving:	nieuw		
		analist:	Andries van der Velde		
		navigator:	<< naam navigator >>		

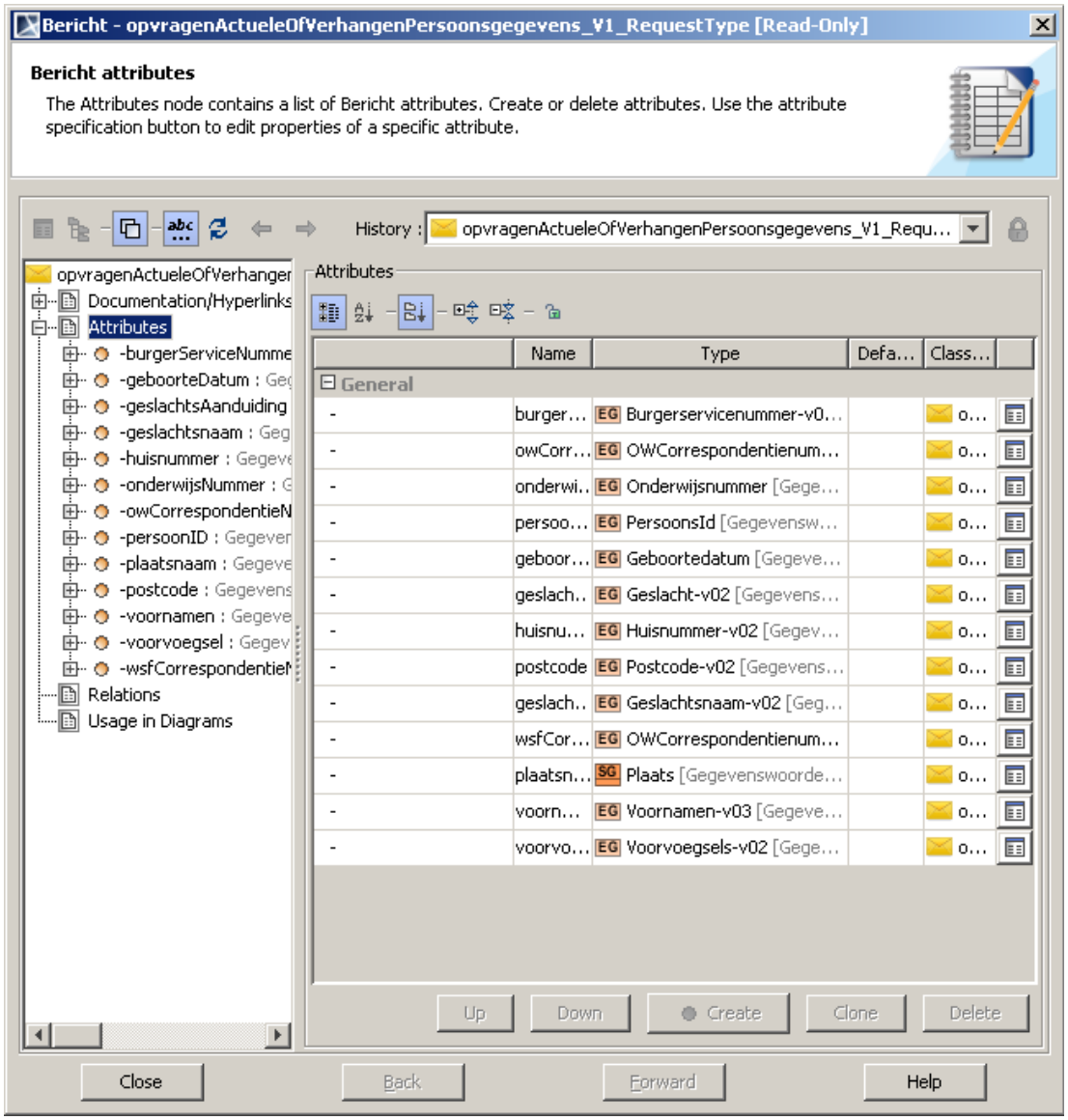
Figuur D.3: Voorbeeld van een met actiewoorden beschreven test waarbij de adresgegevens van een klant via een service worden opgevraagd.

D.3 MagicDraw

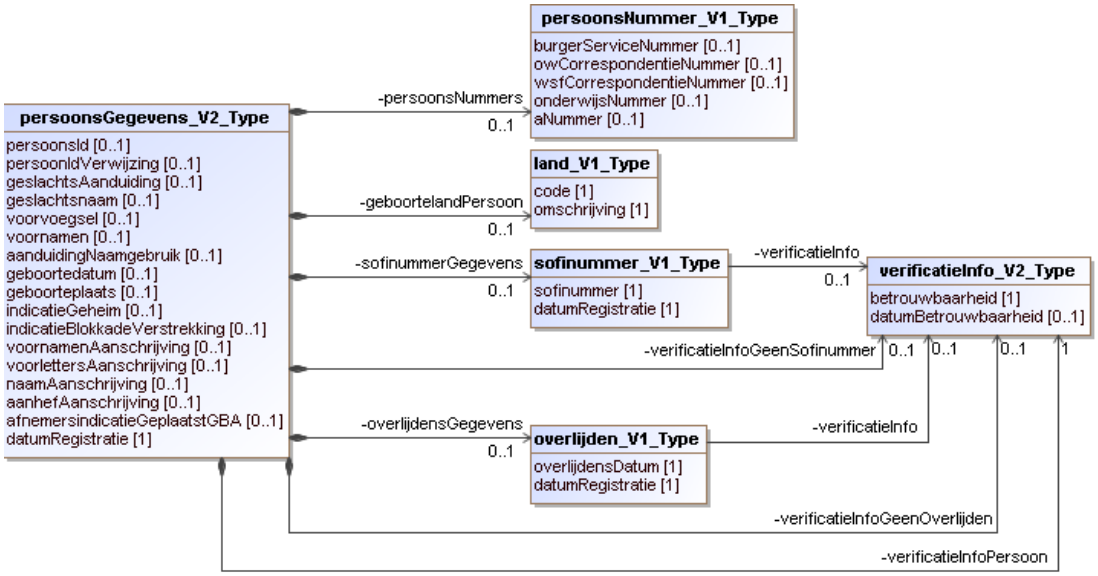
MagicDraw is de UML-modelleringstool waarin onder meer het service- en berichtenboek wordt bijgehouden. Met behulp van de teamworkserver zijn alle gegevens centraal opgeslagen. De figuren D.4 tot en met D.7 tonen enkele screenshots van MagicDraw met betrekking tot de BAP (Basis Administratie Persoon).



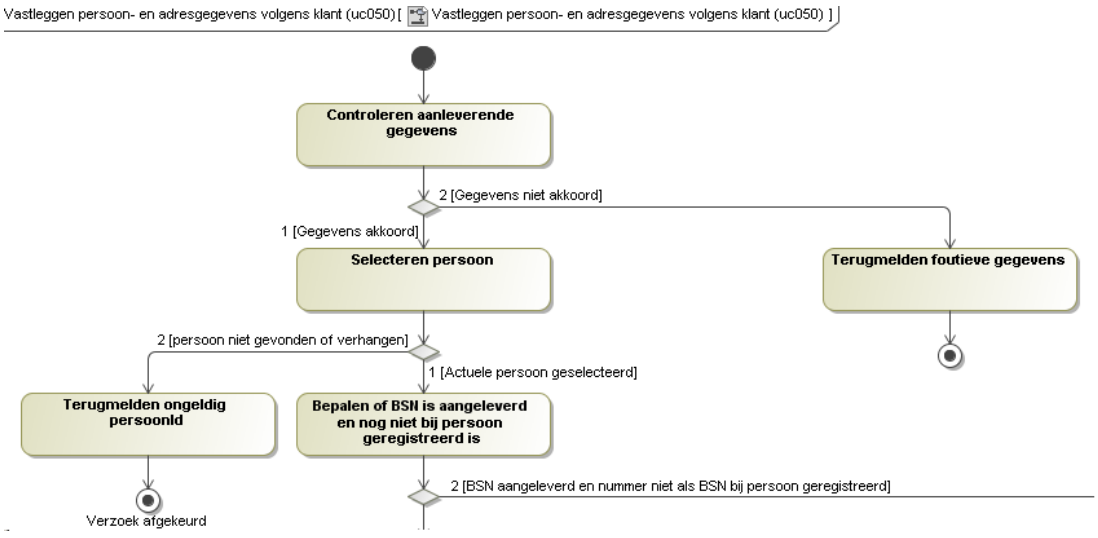
Figuur D.4: Het service- en berichtenboek in MagicDraw. Het opvragen van persoonsgegevens is te vinden als BAP-service. De legacyservices zijn opgenomen als *BasicServices*.



Figuur D.5: De attributen van de *opvragenActueleOfVerhangenPersoonsgegevens* service van de BAP.



Figuur D.6: MagicDraw-schema die de afhankelijkheden van het *persoonsGegevens* object toont. Dat object wordt gebruikt bij het opvragen van de gegevens uit BAP.



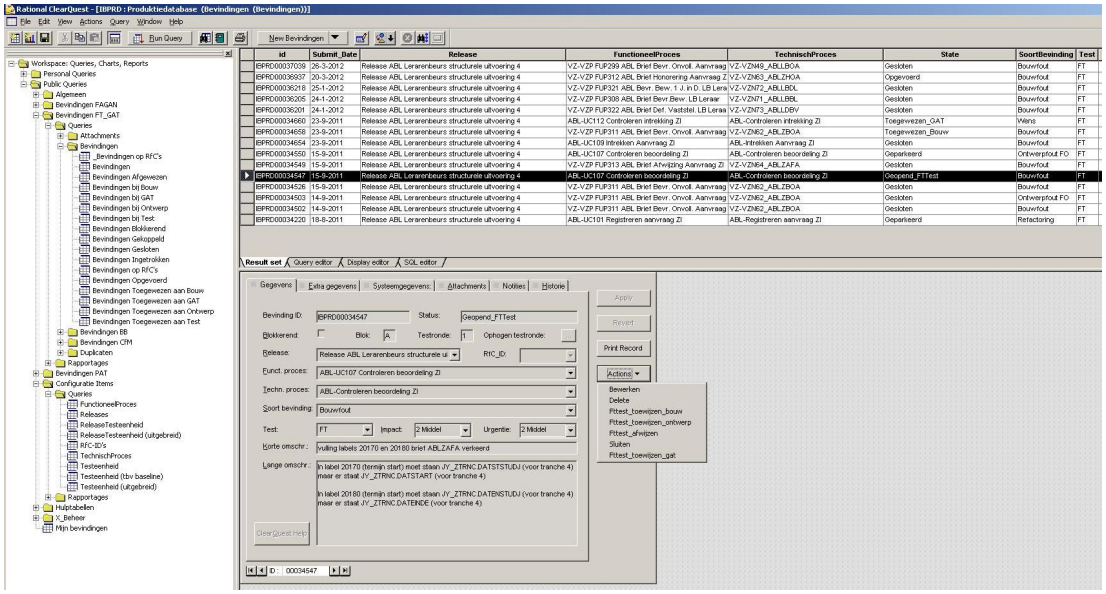
Figuur D.7: MagicDraw-schema van een deel van het proces dat wordt uitgevoerd bij het vastleggen van persoon- en adresgegevens die door een klant zijn aangeleverd.

D.4 Rational ClearQuest

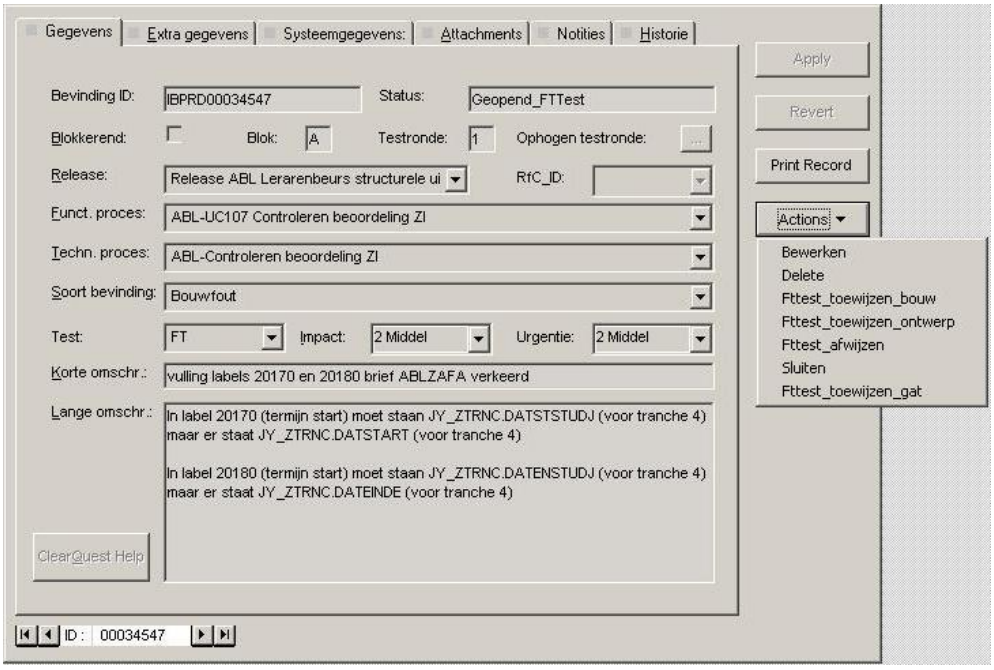
ClearQuest wordt gebruikt voor het vastleggen van bevindingen. Figuur D.8 toont een screenshot van de tool. Hierbij is te zien dat het scherm is opgesplitst in drie onderdelen:

- **Workspace (links):** Via de workspace kunnen bevindingen worden opgezocht en rapportages worden gemaakt op basis van queries. Er is een onderdeel met vooraf gedefinieerde queries (Public Queries) en een onderdeel waar eigen queries kunnen worden toegevoegd (Personal Queries).
- **Query-editior (rechtsboven):** Bevat afhankelijk van het gekozen tabblad de resultaten van een uitgevoerde query, de query editor (voor het toevoegen of wijzingen van filters) of de display editor (voor het kiezen van velden en de sortering).
- **Detailgegevens (rechtsonder):** Toont de detailgegevens van het gekozen item uit de resultatenset. Via dit onderdeel is het mogelijk om wijzigingen uit te voeren of een status te veranderen met behulp van de actie knop. Een vergroting van deze detailweergave staat in figuur D.9. In het geval van een rapportage wordt het rapport hier getoond met de de mogelijkheid om deze te exporteren naar diverse formaten.

Iedere bevinding is voorzien van een soort. In tabel D.1 zijn de verschillende soorten bevindingen met bijbehorende omschrijving beschreven.



Figuur D.8: Een overzicht van ClearQuest met links de workspace, rechtsboven de query-editor en rechtsonder de detailgegevens.



Figuur D.9: Een in ClearQuest geregistreerde bevinding van het type "Bouwfout".

Bevinding	Omschrijving
Bouwfout	Programma is niet conform het functioneel ontwerp. Het toepassen van de checklist kan deze fout niet voorkomen.
Bouwtestfout	Fouten die de bouwer had kunnen voorkomen met behulp van de checklist.
Functionele integratiefout	Functionele ontwerpen zijn niet op elkaar afgestemd.
Omgevingsfout	De uitgangssituatie voor het uitvoeren van de test is niet juist. De omgeving is niet juist geconfigureerd.
Ontwerpfout FO	De fout is te wijten aan een onjuiste beschrijving in het functioneel ontwerp.
Ontwerpfout FO	De fout is te wijten aan een onjuiste beschrijving in het technisch ontwerp. Deze constatering kan niet door een tester worden gedaan, maar alleen door een technisch ontwerper of bouwer.
Overzetfout	Een fout veroorzaakt door een niet goed of volledig verlopen overzetting.
Specificatie-wijziging	De software is conform het functioneel ontwerp gebouwd, maar doet niet wat de gebruiker wil (kan alleen worden gebruikt voor de GAT).
Technische integratiefout	De communicatie tussen de verschillende onderdelen zijn niet juist op elkaar afgestemd.
Testfout	Er bevind zich een fout in de testset.
Wens	De geconstateerde beperking is niet in het functioneel ontwerp beschreven. Er zal geen aanpassing plaatsvinden.

Tabel D.1: Overzicht van de soorten bevindingen die in ClearQuest kunnen worden geregistreerd.

E Concept ICT-principes

E.1 Algemene ICT-principes

Tabel E.1 bevat de algemene ICT-principes van DUO. Deze staan op de architectuurwiki van DUO beschreven. Er moet worden opgemerkt dat alle principes zich in de conceptfase bevinden.

ID	Omschrijving
PAL-01	**SOA en EDA als basis** SOA en EDA (Event Driven Architectuur) vormen de belangrijkste basis van de applicatiearchitectuur.
PAL-02	**Onderscheid tussen primaire en secundaire processen** Er wordt onderscheid gemaakt tussen applicaties ter ondersteuning van het primaire proces en die van secundaire (meestal bedrijfsvoerings) processen.
PAL-03	**Best-fit oplossingen** Processen worden ondersteund met best-fit oplossingen.
PAL-04	**Enkelvoudig gegevensbeheer en meervoudig gebruik** Gegevens worden op één plek beheerd en vaker gebruikt.
PAL-05	**Verwevenheid is minimaal** Oplossingen zijn zodanig ontwikkeld of gekozen dat ze niet afhankelijk zijn van specifieke keuzes in andere onderdelen van de applicatiearchitectuur. Dit geldt zowel voor onderliggende lagen, waarvan het duidelijkste voorbeeld platformonafhankelijkheid is, als voor andere applicaties.
PAL-06	**Efficiënt aanpasbare en uitbreidbare oplossingen** Het applicatielandschap is efficiënt aanpasbaar en uitbreidbaar.
PAL-07	**Voldoende markt- en overheidspenetratie** Oplossingen worden in voldoende mate bij overheidspartijen of in de markt gebruikt.
PAL-08	**ICT-componenten worden gedeeld en gestandaardiseerd voor beide vestigingen van DUO** Sinds 1 januri 2010 zijn CFI en IB-Groep gefuseerd tot DUO. Op infrastructuurgebied zijn er echter nog steeds twee gescheiden infrastructuren met elk een eigen beheer verantwoordelijke (ATOS en ICT DUO-G). Het principe stelt voor om aan deze scheiding een eind te maken en te komen tot één gezamelijke gestandaardiseerde infrastructuur met één beheer partij.
PAL-09	**Alle vernieuwingen en aanpassingen zijn gericht op integratie tussen DUO-Z en DUO-G** Bij alle wijzigingen dient rekening gehouden te worden met beide DUO vestigingen en hun ICT inrichting. Dit komt tot uitdrukking in standaardisering op producten en inrichting, het koppelen van systemen en hergebruik van aanwezige voorzieningen op de andere vestiging.

ID	Omschrijving
PAL-10	**Alle systemen worden vanuit het datacenter in Groningen aangeboden** De systemen van DUO Zoetermeer staan momenteel in het rekencentrum van ATOS in Eindhoven. De systemen van DUO Groningen staan in 2 rekencentra in Groningen onder eigen beheer. Dit principe bepaalt dat alle infrastructuur voor primaire DUO Zoetermeer systemen ook vanuit Groningen wordt aangeboden.
PAL-11	**7x24 beschikbaarheid** Om alle ICT-gerelateerde dienstverlening 7x24 aan te kunnen bieden, moeten grote delen van de ICT-infrastructuur 7x24 uur beschikbaar zijn.
PAL-12	**De infrastructuur is modern, state-of-the-art en loopt, waar nodig, bewust voorop** De infrastructuur is state-of-the-art en loopt (samen met de business) bewust voorop. De infrastructuur moet excellent functioneren en met moderne technieken opgebouwd zijn. Het kennisniveau van de betrokken medewerkers moet daar op aansluiten.
PAL-13	**Alle ICT-componenten zijn web-georiënteerd** Alle hard en softwarecomponenten worden conform webstandaarden, web-best-practices en via web-interfaces ontsloten c.q. gebruikt.
PAL-14	**Per functionaliteit één oplossing gekoppeld aan een strak versie- en licentiebeheer** Per functionaliteit wordt één oplossing en daarvan één versie geïmplementeerd of in stand gehouden. Per oplossing kan dat een OTS (Off-The-Shelf)-pakket of maatwerk zijn.
PAL-15	**De logische werkplek-voorziening is tijds-, plaats- en apparaatonafhankelijk** De logische werkplek-voorziening moet losgekoppeld worden van het onderliggende platform, zodanig dat het op afstand te gebruiken is, op verschillende apparaten, ook buiten kantoortijden.
PAL-16	**Data wordt enkelvoudig opgeslagen en meervoudig gebruikt** Data wordt slechts op één plaats in het netwerk opgeslagen. Een specifiek document (of databaserecord) komt daarom nooit meer dan eens voor.
PAL-17	**Systemen voor secundaire processen sluiten waar mogelijk aan op rijksbrede initiatieven** Veel (zo niet alle) secundaire processen zijn niet DUO-specifiek. Om deze reden vinden er buiten DUO velerlei projecten plaats die secundaire (PIOFA-) processen (gecentraliseerd) aanbieden. Wanneer deze processen van overheidswege centraal worden aangeboden, neemt DUO daaraan deel.
PAL-18	**ICT-componenten zijn op standaard wijze gekoppeld** Waar koppelvlakken van toepassing zijn (logisch, fysiek, virtueel, informatie) wordt dit, waar van toepassing, gedaan met open standaarden en uitgevoerd op een (defacto) standaard wijze.

ID	Omschrijving
PAL-19	**Er worden maximaal twee versies tegelijkertijd ondersteund** Bij DUO is een verscheidenheid aan softwarepakketten en OS'en (Operating Systemen) in gebruik. Met het voortschrijden van de tijd, worden upgrades van deze pakketten in gebruik genomen (behoudens uitfasering). DUO wil niet, naast de upgrade van een pakket, ook nog oude versies in gebruik houden.

Tabel E.1: Overzicht van de algemene concept ICT-principes die bij DUO zijn opgesteld.

E.2 Softwareprincipes

Tabel E.2 bevat de softwareprincipes van DUO. Deze staan op de architectuurwiki van DUO beschreven. Er moet worden opgemerkt dat alle principes zich in de conceptfase bevinden.

ID	Omschrijving
PSW-01	**Platform onafhankelijk** Software wordt zodanig geschreven dat het niet afhankelijk is van toevallige keuzes van platformen of andere technische keuzes. Afhankelijkheden die wel toegstaan zijn worden expliciet vastgelegd (bijvoorbeeld in toolbeleid).
PSW-02	**Robuuste software** Software wordt expliciet uitgebreid met maatregelen om robuuste oplossingen te creëren. Voorbeelden hiervan zijn: transactiemanagement (rollbacks etc), maar ook automatische retries bij failures of niet beschikbaar zijn van services etc.
PSW-03	**Gevalsgewijze verwerking** Aanvragen, mutaties en andere (mogelijk externe) aanleidingen worden direct en per geval verwerkt. Ook de gevolgen hiervan worden zo mogelijk direct doorgerekend. Dit wordt ook wel STP (Straight Through Processing) genoemd.
PSW-04	**Multitier architectuur** Applicaties worden gebouwd in verschillende lagen. Deze lagen zijn van elkaar gescheiden middels interfaces. Binnen een applicatie worden in elk geval de volgende onderdelen ontkoppeld: Webschermen (layout, interactie met gebruiker), webserviceontsluiting (XML-parsen, XSD controles), servicelayer (businessfunctionaliteit, transactieeenheid), DAO (abstractielagen van database).
PSW-05	**Scheiden van interface en implementatie** Van herbruikbare stukken code (zie ook PSW-04) wordt de interface (datgene wat de gebruiker van de oplossing ziet) gescheiden van de implementatie (hiding principe).

ID	Omschrijving
PSW-06	**Asynchroon koppelen** Koppeling worden - indien mogelijk - asynchroon vormgegeven. Hiermee wordt bedoeld dat de de afnemer een koppeling moet kunnen leggen naar een ander onderdeel zonder dat gewacht hoeft te worden op het antwoord. Het aangeroepen onderdeel hoeft niet eens beschikbaar te zijn om toch te kunnen koppelen. Standaard wordt dit geïmplementeerd middels messaging (JMS).
PSW-07	**Transacties niet over servicegrenzen heen** Een service levert een afgeronde eenheid van werk op. Na een serviceaanroep waarin een mutatie is doorgevoerd is deze mutatie onherroepelijk en kan dus niet technisch (bijvoorbeeld middels een rollbackmechanisme) teruggedraaid worden.

Tabel E.2: Overzicht van de concept softwareprincipes die bij DUO zijn opgesteld.

E.3 Applicatieprincipes

Tabel E.3 bevat de applicatieprincipes van DUO. Deze staan op de architectuurwiki van DUO beschreven. Er moet worden opgemerkt dat alle principes zich in de conceptfase bevinden.

ID	Omschrijving
PAP-01	**Hergebruik van componenten, patterns en aspecten** Hergebruik is natuurlijk gebaseerd op runtime componenten maar daarnaast ook op basis van patterns en aspects.
PAP-02	**Hergebruik matchen op FR en NFR** Een oplossing is pas een herbruikbare oplossing als het zowel voldoet aan de functionele als niet functionele eisen.
PAP-03	**Oplossingen volledig gebruiken** Streef naar een oplossing die niet meer functionaliteit biedt dan vereist.
PAP-04	**Oplossingen zonder overlap** Streef naar een oplossing die niet overlapt met bestaande functionaliteit.
PAP-05	**Service Oriented Architecture** (Her)bruikbare services zijn de belangrijkste pijler onder de applicatie architectuur. Alle generieke en herbruikbare functionaliteit wordt middels services ontsloten.
PAP-06	**Legacy koppelen middels services** Legacy systemen worden binnen het SOA systeemlandschap hergebruikt door middel van het verservicen van onderdelen ervan.

ID	Omschrijving
PAP-07	**Integratie voor duplicatie** Centrale gegevens worden zoveel mogelijk via services geïntegreerd met de afnemer. Pas als blijkt dat dit absoluut niet haalbaar is, worden de gegevens gedupliceerd naar een omgeving waar de afnemer wel goed bij kan.
PAP-08	**Horizontale schaling** De capaciteit (verwerkingshoeveelheden) van applicaties neemt lineair toe bij het parallel bijschakelen van meer hardware (in plaats van het vervangen van huidige door snellere hardware).
PAP-09	**Audit trail** Elke applicatie is zelf verantwoordelijk voor het onomstotelijk vastleggen van die gegevens die het mogelijk maken om (eventueel later) een correcte audittrail op te bouwen over de door de applicatie gewijzigde gegevens. Deze (meta-)gegevens worden in de gegevensregistraties zelf vastgelegd of direct daaraan gekoppeld.
PAP-10	**Self Service Security** Elk component is verantwoordelijk voor zijn eigen beveiliging.
PAP-11	**(Near-)zero maintenance** Applicaties worden ontwikkeld worden met in het achterhoofd dat ze goed mogelijk blijven draaien met een minimale beheerlast. Gevalsgewijze verwerking die om technische redenen fout is gegaan moet bijvoorbeeld automatisch opnieuw aangeboden kunnen worden. Functionele fouten worden zoveel mogelijk automatisch afgehandeld.
PAP-12	**Onderscheid tussen specifieke gebruikers- en herbruikbare generieke oplossingen** Er wordt onderscheid gemaakt tussen specifieke gebruikers-gerichte applicaties en herbruikbare generieke applicaties (bijvoorbeeld registerapplicaties).

Tabel E.3: Overzicht van de concept applicatieprincipes die binnen DUO zijn opgesteld.

E.4 Infrastructuurprincipes

Tabel E.4 bevat de infrastructuurprincipes van DUO. Deze staan op de architectuurwiki van DUO beschreven. Er moet worden opgemerkt dat alle principes zich in de conceptfase bevinden.

ID	Omschrijving
PIN-01	**DUO heeft 2 redundante rekencentra** De Disaster-Recovery uitwerking van DUO bestaat uit het gebruik maken van twee gelijkwaardige rekencentra. Bij volledige uitval van één van de twee rekencentra is de continuïteit van de ICT voorzieningen gewaarborgd.

ID	Omschrijving
PIN-02	**Bij calamiteit mag maximaal 24 uur dataverlies optreden** Bij een calamiteit, bijvoorbeeld uitval van een geheel rekencentrum, mogen van maximaal 24 uur de mutaties in de geautomatiseerde systemen verloren gaan.
PIN-03	**Alle ICT componenten worden gevirtualiseerd** Alle ICT hardware en software componenten worden gevirtualiseerd als de techniek er geschikt voor is. Hardware en software componenten worden hiermee ontkoppeld opgezet. Het OS wordt losgekoppeld van de hardware(virtualisatie), en software wordt losgekoppeld van het OS of platform(applicatie virtualisatie).
PIN-04	**De infrastructuur bestaat uit gestandaardiseerde bouwblokken** Gestandaardiseerde infrastructuur bouwblokken bevorderen de interoperabiliteit (koppeling / uitwisseling digitale informatie). Met behulp van standaard bouwblokken vereenvoudigt de infrastructuur en communicatie tussen (semi-) overheden onderling en tussen overheid, bedrijven en burgers.
PIN-05	**Wanneer er geen rijksbrede initiatieven zijn, heeft het datacenter van DUO de voorkeur** Vaak kan DUO aansluiten bij rijksbrede initiatieven voor (met name ondersteunende, PIOFA)processen. Wanneer dit niet kan, ontwikkelt DUO een eigen oplossing. Dit kan middels pakket (evt. cloud), eigen ontwikkeling of een combinatie. Deze nieuwe oplossing wordt, voor zover het gaat om eigen beheer, in de DUO-eigen rekencentra geplaatst, uitgezonderd de diensten/resources die (al of niet via het internet) van andere leveranciers afgenomen worden.

Tabel E.4: Overzicht van de concept infrastructuurprincipes die bij DUO zijn opgesteld.

F Leerpuntenrapport

Na het afstudeeronderzoek is een leerpuntenrapport opgesteld. Dit rapport bevat punten die tijdens het onderzoek zijn opgevallen en voor een vervolgonderzoek nuttig kunnen zijn. Door deze informatie te delen met anderen kan tijd worden bespaard en worden herhaaldelijke knelpunten voorkomen.

De volgende leerpunten zijn tijdens dit onderzoek naar voren gekomen:

- **Technische voorzieningen**: Er zou veel tijd bespaard kunnen worden door direct bij aanvang een zogeheten 'statefull werkplek' te krijgen. Deze werkplekken bieden administratorrechten en meer CPU-kracht en geheugen. In het begin van het onderzoek is tijd verloren gegaan door work-arounds te vinden voor het installeren van onder andere LaTeX- en GIT-software. Ook duurde het genereren van een PDF-document uit een LaTeX-bestand (dat tientallen keren op een dag gebeurd) door de beperkte CPU-kracht erg lang.
- **Rijkspas eerder aanvragen**: De Rijkspas is een toegangspas voor gebouwen van de Rijksoverheid. Het aanvragen van deze pas kan enkele weken duren. Wanneer de Rijkspas nog niet beschikbaar is, moet elke ochtend een dagpas worden aangevraagd bij de receptie. Dit kan door de wachtrijen soms erg lang duren, waardoor kostbare tijd verloren gaat. De Rijkspas zou het beste vóór de start van het afstudeeronderzoek aangevraagd kunnen worden, zodat deze op de eerste werkdag klaar ligt en direct gebruikt kan worden.
- **Contractduur vaststellen**: Het afstudeeronderzoek duurde tot 6 juli 2012. Het contract was echter getekend tot 31 juni 2012. Na deze datum is er geen toegang meer tot de systemen van DUO, tenzij het contract wordt verlengd. Het is aan te raden om de contractdatum bij het begin goed vast te stellen of om ruim op tijd een contractverlenging aan te vragen. Het duurt enige tijd voordat een contractverlening in alle systemen correct is doorgevoerd.
- **Tweede begeleider eerder regelen**: Het kan geen kwaad om direct aan het begin van het onderzoek een tweede begeleider vanuit de universiteit te vinden. Tijdens dit onderzoek is dit pas halverwege gedaan. Het pakte gelukkig goed uit: de feedback was positief en er waren geen schokkende wijzigingen nodig. Er bestaat echter altijd de kans dat dit wel gebeurd. Ingrijpende wijzigingen zijn halverwege een onderzoek altijd vervelender dan wanneer dit in het beginstadium zou plaatsvinden.
- **Feedbackmomenten aankondigen**: Door van tevoren aan te kondigen wanneer een conceptversie wordt opgeleverd kunnen de partijen die feedback gaan geven hier rekening mee houden en alvast tijd inplannen. Tijdens het eerste feedbackmoment was dit niet op tijd gedaan, waardoor het langer duurde voordat de feedback werd gegeven.