Afstudeerverslag

Predicting Server Flaws

Anthony Huebers

Afstudeerverslag

Predicting Server Flaws

Gegevens	Student
Naam	Anthony Huebers
Studentnummer	2182775
Profiel	ICT & Software (Voltijd)
Afstudeerperiode	31-08-2015 t/m 03-02-2016

Gegevens	Bedrijf
Naam	Info Support B.V.
Afdeling	Professional Development Center
Plaats	Veenendaal
Bedrijfsbegeleider	Tom Nieuwenhuijs (Software Engineer)
Getekend voor gezien	

Gegevens	Docentbegeleider
Naam	Jelle Oosterkamp

Titel	Afstudeerverslag
Project/Onderwerp	Predicting Server Flaws
Versie	1.3
Status	Definitief
Datum	07-jan-2016
Bestand	Afstudeerverslag - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM, Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	19-11-2015	Anthony Huebers	Creatie
1.1	Concept	23-12-2015	Anthony Huebers	Feedback Hylke Peek en Tom Nieuwenhuijs verwerkt
1.2	Definitief	24-12-2015	Anthony Huebers	Verslag compleet gemaakt en bijlages toegevoegd
1.3	Definitief	07-01-2016	Anthony Huebers	Feedback Hylke Peek, Tom Nieuwenhuijs en Jelle Oosterkamp verwerkt

Distributie

Versie	Status	Datum	Aan
1.0	Concept	16-12-2015	Tom Nieuwenhuijs & Hylke Peek
1.2	Definitief	24-12-2015	Tom Nieuwenhuijs, Hylke Peek & Jelle Oosterkamp
1.3	Definitief	07-01-2016	Tom Nieuwenhuijs, Hylke Peek & Jelle Oosterkamp

Voorwoord

Dit document is geschreven ter afronding van de opleiding ICT & Software aan de Fontys Hogescholen te Eindhoven. In dit voorwoord wil ik graag iedereen bedanken die mij tijdens het afstuderen heeft begeleid of ondersteund.

De begeleiders van Info Support hebben mij op zowel technisch als procesmatig gebied goed begeleid. Hiervoor wil ik in het bijzonder Tom Nieuwenhuijs, Gert Jan Timmerman en Hylke Peek en Marieke Keurntjes bedanken. Daarnaast wil ik mijn kamergenoten en medeafstudeerder Simon de Lang bedanken voor het luisteren naar mijn verhalen en de goede sfeer.

Vanuit Fontys Hogescholen wil ik graag Jelle Oosterkamp en Jos Boonen bedanken. Jelle Oosterkamp heeft mij gedurende de afstudeerstage begeleid om tot een succesvolle afronding van mijn afstudeerstage te komen. Daarnaast heeft Jos Boonen mij gedurende het driejarig studietraject goed begeleid.

Tot slot wil ik graag mijn vriendin Lindsey Jongen, schoonzus Laura Jongen en mijn ouders John en Monique Huebers bedanken voor hun steun en het geven van feedback op spelling en grammatica van documenten.

Management samenvatting

Dit document is geschreven ter afronding van de opleiding HBO ICT & Software aan de Fontys Hogescholen te Eindhoven. Gedurende de afstudeerstage van september 2015 tot en met februari 2016 is het project "Predicting Server Flaws" uitgevoerd bij Info Support.

Dit project is gestart omdat het regelmatig voorkomt dat er onverwachts een probleem ontstaat op een (machine op een) omgeving (van klanten) die beheerd wordt door Info Support. Als op voorhand gesignaleerd kan worden wanneer een omgeving problemen dreigt te krijgen, zou een werkelijk probleem mogelijk voorkomen kunnen worden.

Het oorspronkelijk gewenste resultaat was een (proof of concept) systeem wat (patronen in) problemen kan herkennen en mogelijk problemen in de toekomst kan voorspellen. Het systeem zou mogelijke problemen alleen signaleren en niet daadwerkelijk oplossen voor deze ontstaan.

Tijdens de eerste drie weken van dit project is een Plan van Aanpak opgesteld en hebben er meerdere (kennismakings)gesprekken met alle betrokkenen plaatsgevonden om de opdracht helder te krijgen.

Na het afronden van het Plan van Aanpak is een onderzoek uitgevoerd. Uit het onderzoek bleek dat het maken van voorspellingen van problemen die gekoppeld zijn aan gegevens uit systeem-, beschikbaarheids- en reactie informatie mogelijk is. Door middel van Time Series Analysis kan het ontstaan van afwijkend gedrag voor gegevens uit systeem-, beschikbaarheids- en reactie informatie voorspeld worden. Dit is een algoritme om tijd gedreven gegevens mee te analyseren. Door de verwachting te vergelijken met het daadwerkelijke gedrag wordt afwijkend gedrag bekend voordat problemen ontstaan.

Tijdens het onderzoek is er ook aandacht besteed aan het zoeken en vergelijken van beschikbare softwarepakketten voor het verkrijgen en centraliseren van de benodigde gegevens. Ook zijn de centrale opslagmogelijkheden binnen NoSQL onderzocht. Omdat aan de hand van systeem-, beschikbaarheids- en reactie informatie het beste voorspellingen gemaakt kunnen worden, is het advies hierop de focus te leggen bij de realisatie van een concept.

Na het afronden van het onderzoek is het systeem in meerdere iteraties gerealiseerd middels de Agile Scrum ontwikkelmethodiek. Tijdens de ontwikkeling van het systeem bleek dat de verkregen data niet bruikbaar was voor het voorspellen van problemen en/of afwijkend gedrag. Om de doelstelling tot proactief handelen te verwezenlijken is daarom besloten (de totale processor) verbruiken te voorspellen. Het voorspellen van verbruiken kan resulteren in het voorspellen van afwijkend gedrag dat ontstaat door hoge of afwijkende verbruiken. Er kan namelijk worden voorspeld dat er voor een x periode een y verbruik gaat plaatsvinden. Of en hoe afwijkend gedrag gerelateerd is aan een of meerdere verbruiken is nu echter nog onbekend.

Door de toepassing van de Agile Scrum ontwikkelmethodiek is het echter niet erg dat het eindresultaat afwijkt van het oorspronkelijk gewenste resultaat. Gedurende de ontwikkelfase is er namelijk voor iedere sprint in samenspraak met de opdrachtgever een resultaat bepaald. De afgestemde resultaten werden aan het einde van iedere sprint behaald.

Het opgeleverde systeem bewijst dat het mogelijk is realtime gegevens te verkrijgen, bewaren, te analyseren en daar vervolgens realtime voorspellingen mee te maken. De integratie van dit proces en dan met name de integratie van de statistische programmeertaal R en .NET is waardevolle kennis. De verkregen gegevens en gemaakte voorspellingen zijn inzichtelijk gemaakt in een dashboard.

Op technisch en theoretisch gebied is veel mogelijk en er wordt daarom aanbevolen een vervolg onderzoek en of project te starten wanneer er geclassificeerde gegevens beschikbaar zijn. De in dit project toegepaste technieken hebben bewezen dat het technische mogelijk is realtime gegevens te verkrijgen en te analyseren om hier vervolgens realtime voorspellingen mee te maken.

Management summary

This document was written in order to complete the HBO ICT & Software education at Fontys Hogescholen in Eindhoven. During the graduation internship from September 2015 until February 2016, the project "Predicting Server Flaws" was conducted at Info Support.

This project was started because unexpected problems emerge regularly at (a machine at) an environment (of customers) that is managed by Info Support. By signaling in advance when an environment is likely to experience problems, these problems could possibly be prevented.

The initially intended result was a (proof of concept) system that is able to recognize (patterns in) problems and as such predict future problems. The system should only signal potential problems and should not solve problems before they emerge.

During the first three weeks of this project a "Project Proposal" was written and multiple (introductory) interviews were conducted with all stakeholders in order to get the project clear.

After completing the "Project Proposal", research was conducted. The research showed the possibility of predicting problems linked to system-, availability- and response information. Through Time Series Analysis, the emerging of deviant behavior linked to system, availability and response information can be predicted. This is an algorithm for analyzing time driven data. Predictions are made by comparing the expected and actual behavior to identify deviant behavior.

During the research, attention was spent on searching and comparing existing software packages in their ability to obtain and centralize the required data. The central storage possibilities within NoSQL have also been researched. Due to the possibility to predict problems linked to system-, availability- and response information, it is highly recommended to keep this in mind during the realization of a concept.

After completing the research, the system was created in several iterations through the Agile Scrum development methodology. Throughout the development of the system it became clear that the obtained data was not usable for predicting problems or deviant behavior. In order to achieve the goal to act proactively it was therefore decided to predict (the total processor) consumption. By predicting consumptions, it is possible to predict deviant behavior which arises from high or deviant consumptions. For it is possible to predict the future x consumption for a y period. Whether and how deviant behavior is related to one or more consumptions is still unknown.

Due to the use of the Agile Scrum development methodology it does not matter that the end result differs from the initially intended result. The goal of each iteration was determined in collaboration with the client. At the end of each sprint the determined goal was reached.

The created system proves that it is possible to obtain, store and analyze data (in real-time) to then make real-time predictions with it. The integration of this process and in particular the integration of the statistical programming language R, and .NET was valuable knowledge. The obtained data and made predictions provide insight in a dashboard.

Due to the technical and theoretical possibilities further research is recommended when classified data becomes available. The technologies used in this project prove that it is technically possible to obtain and analyze data (in real-time) to then use it for real-time predictions.

Verklarende woordenlijst

Afwijkend gedrag
Gedrag dat afwijkt van het gewenste gedrag.

Algoritme
Reeks instructies om vanaf een beginpunt een bepaald doel te bereiken *(Kernerman Dictionaries, z.d.)*. Deze reeks van instructies zijn vaak wetenschappelijk onderbouwd en hebben zich in het verleden meestal bewezen.

Business Intelligence (BI)
Staat voor het verzamelen en structureren van bedrijfsgegevens zodat hieruit informatie wordt gewonnen voor het beheren en bijsturen van processen *(Marqit, z.d.)*.

Dependency injection
Een ontwerppatroon uit de informatica dat het mogelijk maakt programmacomponenten losjes te koppelen zonder dat deze relatie hard in de programmacode vastgelegd is *(Wikipedia, z.d.)*.

Logbestand
Een tekstbestand waarin staat beschreven welke evenementen er hebben plaatsgevonden *(Christensson, 2010)*.

Logbericht
De beschrijving van een evenement.

Machine learning
Een studiegebied dat computers de mogelijkheid biedt van gegevens te leren zonder dat expliciet te programmeren *(Munoz, z.d.)*.

NoSQL
Ook wel “Not only SQL”, verzamelnaam voor niet-relationele databasesystemen *(Khan Zaki, 2014)*.

Observatie
Een of meerdere bij elkaar horende metingen voor een specifiek moment in tijd.

Probleem
Wanneer een systeem of applicatie anders functioneert dan gewenst.

Softwareframework
Een verzameling universele functionaliteiten, een architectuur en/of richtlijnen en afspraken die gebruikt kunnen worden om snel en gestructureerd applicaties te ontwikkelen *(Clifton, 2003)*.

Systeeminformatie
Allesomvattend woord voor informatie over de hardware (zoals de temperatuur of het gebruik) en de softwareomgeving (zoals het besturingssysteem of de naam van het systeem) van een (machine op een) omgeving.

Verbruiken
De (percentuele) belasting van de processor, het geheugen of de hardeschijf per tijdseenheid.

Voorspelling
Het doen van uitspraken omtrent toekomstige gebeurtenissen en ontwikkelingen binnen een systeem, op grond van het historisch gedrag van dat systeem *(Woorden-boek.nl, z.d.)*.

Inhoudsopgave

1. Inleiding

Dit verslag beschrijft de aanpak, uitvoering en resultaten van de afstudeerstage met het project "Predicting Server Flaws" uitgevoerd door Anthony Huebers voor Info Support B.V. en Fontys Hogescholen te Eindhoven. Het doel van dit verslag is het inzichtelijk maken van het doorlopen afstudeerproces.

1.1 Leeswijzer

Het verslag begint met een beschrijving van de achtergrond en de opdracht. In hoofdstuk 2 wordt de achtergrond beschreven te beginnen met een beschrijving van het bedrijf waarvoor de project is uitgevoerd. Vervolgens worden de betrokkenen bij dit project benoemd. In hoofdstuk 3 wordt de opdracht beschreven. In de beschrijving van de opdracht wordt de aanleiding, doelstelling, probleemstelling, het gewenste resultaat en effect benoemd. Daarna wordt het onderzoeksaspect toegelicht met daarin de gestelde hoofdvraag en deelvragen.

De aanpak is beschreven in hoofdstuk 4. Als eerste worden de toegepaste methodieken per fase beschreven. Vervolgens worden de technieken beschreven die gedurende de uitvoering van het project gebruikt zijn. Daarna wordt de planning en de afwijkingen op de planning beschreven.

De uitvoering en resultaten van het project worden besproken in hoofdstuk 5. Hierin is het volledige afstudeerproces beschreven met daarin de voortgangsrapportages, het Plan van Aanpak, de uitvoering en resultaten per fase en overige activiteiten.

Uit de uitvoering van het project kunnen verschillende conclusies worden getrokken en aanbevelingen worden gemaakt. Deze staan beschreven in hoofdstuk 6.

Wat ik geleerd heb van dit project en wat ik in de toekomst anders zou doen staan beschreven in hoofdstuk 7.

Dit verslag eindigt met een lijst van gebruikte bronnen in hoofdstuk 8 en de opgenomen bijlagen in hoofdstuk 9. In de bijlagen zijn de opgeleverde deelresultaten en de specificaties van de ontwikkelomgeving terug te vinden.

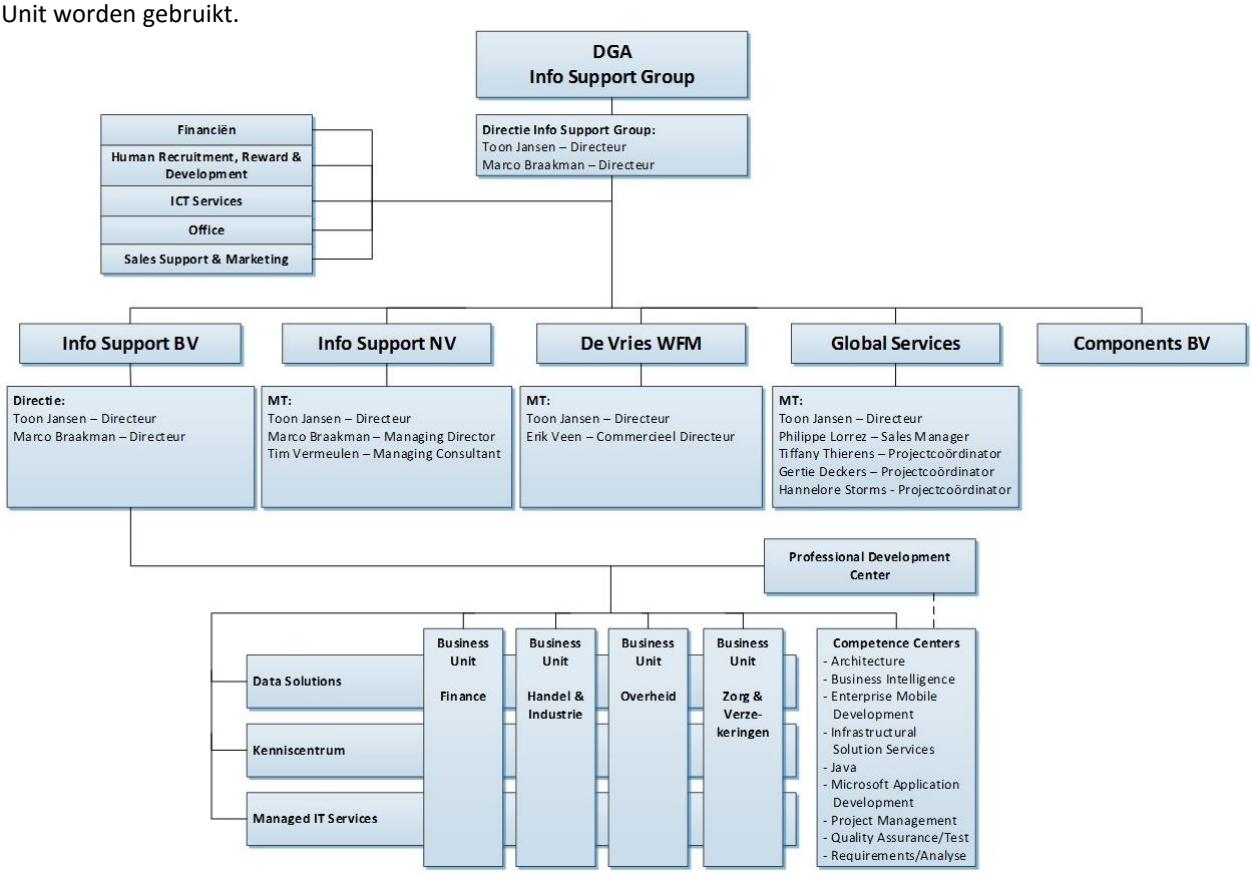
2. Context

2.1 Bedrijfsbeschrijving

Info Support is een specialist in het ontwikkelen, beheren en hosten van software op maat en het leveren van BI- en integratieoplossingen. Het bedrijf bestaat al sinds 1986 en is inmiddels van een eenmansbedrijf uitgegroeid tot ruim 400 medewerkers met vestigingen verspreid over Nederland en België.

In het logo van Info Support is de term 'Solid Innovator' verwerkt. Met solide wordt duurzaam, stevig en betrouwbaar bedoeld. Dit resulteert in langetermijnrelaties en structurele, kwalitatief hoogwaardige oplossingen. Met innovatief wordt het continue investeren in kennis van business en technologische innovaties bedoeld. Innovatie is een keuze die Info Support stimuleert. Door middel van een eigen kenniscentrum worden ruim 300 trainingen, veelal gericht op softwareontwikkeling, verzorgd aan interne medewerkers en aan externe klanten. Zowel het hoofdkantoor als het kenniscentrum zijn gevestigd in Veenendaal *(Info Support, z.d.)*.

In *Figuur 1 - Organogram* is te zien dat Info Support Group verschillende dochterbedrijven bezit. De BV is opgedeeld in vier verschillende Business Units. Dit zijn "afdelingen" die elk gericht zijn op een specifieke bedrijfstak (Finance, Handel & Industrie, Overheid en Zorg & Verzekering). Daarnaast zijn er een aantal faciliterende afdelingen (Professional Development Center, Data Solutions, Kenniscentrum en Managed IT Services) die door iedere Business Unit worden gebruikt.



Figuur 1 - Organogram

2.2 Rollen

Het project is uitgevoerd door de opdrachtnemer. Tijdens de uitvoering van het project heeft de opdrachtnemer ondersteuning gekregen van enkele rollen. Deze rollen zijn benoemd in *Tabel 1 - Rollen binnen Info Support BV* en *Tabel 2 - Rollen binnen Fontys Hogescholen*. Zie voor een uitgebreidere omschrijving *hoofdstuk 4.1 Organisatie* van het Plan van Aanpak, te vinden in bijlage *9.1 Bijlage A - Plan Van Aanpak*.

Rol	Info Support B.V.
Opdrachtnemer	Anthony Huebers
Opdrachtgever	Tom Nieuwenhuijs
Technisch begeleider	Hylke Peek
Procesbegeleider	Marieke Keurntjes
Business unitmanager	Mark Klabbers

Tabel 1 - Rollen binnen Info Support BV

Rol	Fontys Hogescholen
Afstudeerbegeleider	Jelle Oosterkamp
Assessor	René van der Heijden

Tabel 2 - Rollen binnen Fontys Hogescholen

3. Opdrachtdefinitie

3.1 Aanleiding

Info Support beheert IT omgevingen voor verschillende klanten. Deze omgevingen moeten idealiter non-stop beschikbaar zijn. Dit is echter niet altijd het geval. Sommige problemen zorgen er zelfs voor dat de (machine op een) omgeving (tijdelijk) niet meer beschikbaar is. Er kunnen problemen optreden door een groot aantal oorzaken. Als op voorhand gemeld wordt wanneer een (machine op een) omgeving problemen dreigt te krijgen, zou een werkelijk probleem mogelijk voorkomen kunnen worden.

De afdeling genaamd Professional Development Center (PDC) faciliteert en beheert de ontwikkelstraat die door alle ontwikkelaars van Info Support gebruikt wordt. Het project wordt voor de afdeling PDC uitgevoerd.

3.2 Probleemstelling

Het komt regelmatig voor dat er onverwachts een probleem ontstaat op een (machine op een) omgeving. Deze problemen vinden vaak op dezelfde of andere omgevingen opnieuw plaats.

3.3 Doelstelling

De opdrachtgever heeft als doel dat er proactief gehandeld wordt om te voorkomen dat er problemen ontstaan. Door afwijkend gedrag van een (machine op een) omgeving te melden, kunnen medewerkers van Info Support hier op inspelen om problemen mogelijk te voorkomen. Hierdoor moet het aantal problemen dat daadwerkelijk ontstaat verminderd worden.

De onderwijsinstelling en opdrachtnemer hebben als gezamenlijk doel het succesvol afronden van de afstudeerstage.

3.4 Resultaat

Om de doelstelling van de opdrachtgever te kunnen verwezenlijken, moet er een concept komen waarmee bewezen wordt dat problemen voorspeld kunnen worden. Het concept moet worden bewezen in de vorm van een (proof of concept) systeem dat problemen voorspelt. Om problemen te kunnen voorspellen moeten problemen of afwijkend gedrag herkend en gemeld worden. Door herkenning kan voorspeld worden of een probleem zich dreigt voor te doen. Wanneer een (eerder opgedaan) probleem dreigt te ontstaan, moet er een melding gegeven worden. Deze meldingen zijn zichtbaar in een dashboard. Op deze manier worden medewerkers van Info Support geïnformeerd en kunnen ze problemen mogelijk voorkomen.

Enkele voorbeelden van afwijkend gedrag waar voorspellingen opgemaakt kunnen worden:

- Uit systeeminformatie kan herleid worden wanneer een harde schijf vol dreigt te raken.
- Uit (de inhoud van) logbestanden kan herleid worden of een applicatie dreigt te crashen, omdat benodigde resources (zoals bijvoorbeeld een database) slecht of zelfs niet toegankelijk zijn.
- Uit beschikbaarheids- en reactie informatie kan herleid worden of een systeem toegankelijk is en hoe snel het systeem reageert op inkomende verzoeken.

Om de doelstelling van de onderwijsinstelling en opdrachtnemer te kunnen verwezenlijken, moet het resultaat voor de opdrachtgever behaald worden. Daarnaast moeten er nog documenten opgeleverd en verantwoord worden aan Fontys Hogescholen.

3.5 Effect

Wanneer er een bewezen concept beschikbaar is waarmee problemen voorspeld kunnen worden, kan de doelstelling van de opdrachtgever om proactief te handelen en mogelijk problemen te verminderen behaald worden. Het proof of concept kan gebruikt worden om het concept (op langer termijn) te testen.

Wanneer van het concept een systeem gerealiseerd is dat problemen voorspelt, zal er naast de doelstelling van de opdrachtgever ook een neveneffect zijn. Het tijdelijke neveneffect van dit resultaat is dat klanten hierdoor in de toekomst minder last zullen ondervinden van problemen omdat deze door eerdere meldingen sneller opgelost of voorkomen kunnen worden. Klanten zullen daardoor meer tevreden worden over de stabiliteit van de (machine op een) omgeving. Dit neveneffect verdwijnt na een periode omdat klanten dan gewend raken aan de nieuwe stabiliteit.

Wanneer naast het resultaat voor de opdrachtgever ook het resultaat voor de onderwijsinstelling behaald wordt, zal het doel om de afstudeerstage succesvol af te ronden bereikt zijn.

3.6 Onderzoek

Om tot een goed concept te komen waarmee problemen voorspeld kunnen worden, is een onderzoek essentieel. Gedurende dit onderzoek moet bekend worden hoe gegevens verkregen en centraal bewaard kunnen worden. Daarnaast moet duidelijk worden welke informatie er uit gegevens gehaald kan worden en op welke manier deze informatie verkregen kan worden. Verder moet duidelijk worden hoe de toekomst voorspeld kan worden aan de hand van informatie uit het verleden. Om dit allemaal te kunnen beantwoorden is er door de opdrachtnemer een hoofdvraag en deelvragen gedefinieerd.

3.6.1 Onderzoeksvragen

De vraag "*Hoe kunnen aan de hand van (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie problemen voorspeld worden?"* staat in het onderzoek centraal.

Om de hoofdvraag te kunnen beantwoorden zijn de volgende deelvragen opgesteld:

- Hoe verkrijg je (de inhoud van) logbestanden en systeeminformatie van een (machine op een) omgeving op een centrale plek?
- Welk NoSQL oplossing is geschikt voor de centrale opslag van de gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie?
- Hoe herken je afwijkend gedrag uit (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie?
- Hoe kun je aan de hand van afwijkend gedrag problemen herkennen zodat deze voorspeld kunnen worden?

De eerste en tweede deelvraag zijn opgesteld omdat er een concept gerealiseerd moet worden. Om een proof of concept te kunnen maken is de technische kennis die deze deelvragen verschaffen essentieel.

4. Aanpak

Dit project kent drie fases, beginnende met een onderzoeksfase, gevolgd door een ontwikkelingsfase en tot slot een afrondingsfase. Een beschrijving van deze fases is te vinden in *hoofdstuk 3.1 Fasering* van het Plan van Aanpak, te vinden in bijlage *9.1 Bijlage A - Plan Van Aanpak*. De afrondingsfase wordt niet benoemd omdat deze bestaat uit het schrijven van dit verslag en de voorbereiding op het afsluiten van de afstudeerstage.

4.1 Methodieken

4.1.1 Onderzoeksfase

Bij de uitvoering van het onderzoek is het triangulatie-framework gebruikt als hulpmiddel om te bepalen hoe het onderzoek uitgevoerd moet worden. Door verschillende methoden die dit framework biedt te combineren, kunnen onderzoeksvragen beter beantwoord worden *(Oates, 2006)*. De verschillende methoden zijn verdeeld in vijf onderzoeksruimtes. Iedere onderzoeksruimte heeft een eigen aanpak en een bijbehorende verzameling methodieken en technieken. Hieronder staat in het kort beschreven welke focus iedere onderzoeksruimte heeft *(Turnhout, et al., 2013)*:

- **Bieb:** richt zich op het verkrijgen van beschikbare literatuur en het voortbouwen op eerder uitgevoerd onderzoek.
- **Werkplaats:** richt zich op het experimenteren met oplossingen om zo tot nieuwe bevindingen te komen.
- **Showroom:** richt zich op de herbruikbaarheid van oplossingen en het vergelijken van verschillende oplossingen.
- **Laboratorium:** richt zich op het toetsen en meten van oplossingen.
- **Veld:** richt zich op het verdiepen in het toepassingsdomein door middel van contact met betrokkenen personen.

4.1.2 Ontwikkelingsfase

Gedurende de ontwikkelingsfase is een aangepaste versie van Agile Scrum gebruikt als ontwikkelmethodiek. Het element daily standup en de rol Scrum master zijn uit de methodiek gehaald, omdat het systeem door een eenmansteam ontwikkeld werd.

Het product backlog element van Agile Scrum is gebruikt voor het bijhouden van een lijst van eisen en wensen, ook wel product backlog items genoemd, van het te ontwikkelen systeem. Gedurende het project zijn hier wensen door de opdrachtgever aan toegevoegd. De opdrachtgever vervult de zogeheten product owner rol en bepaalt de prioritering van deze lijst.

Het sprint element is gebruikt om in meerdere iteraties het systeem te realiseren. Door in sprints (met een duur van twee weken) te werken, blijft de opdrachtgever nauw betrokken bij de voortgang van het systeem. Bij het begin van iedere sprint is in overleg met de product owner een planning gemaakt welke product backlog items in de betreffende sprint gerealiseerd moeten worden. Halverwege iedere sprint werden het (bijgewerkte) functioneel en technische ontwerp opgeleverd. Aan het eind van iedere sprint werden de nieuwe ontwikkelingen aan het systeem gepresenteerd in de vorm van een demo.

Het sprint review/retrospective element is na iedere sprint gebruikt als evaluatiemoment. Hierin bespraken de opdrachtnemer en opdrachtgever wat inhoudelijk (review) en procesmatig (retrospective) goed en fout ging in de desbetreffende sprint. Eventuele verbeterpunten konden dan meegenomen worden in de volgende sprint.

Het definition of done element is gebruikt om in vast te leggen wanneer resultaten succesvol afgerond waren. De definition of done is te vinden in *hoofdstuk 4.5 Kwaliteit* van het Plan van Aanpak, te vinden in bijlage *9.1 Bijlage A - Plan Van Aanpak*.

Agile Scrum is gekozen als ontwikkelmethodiek, omdat dit een methodiek is waarbij in meerdere iteraties het systeem ontwikkeld wordt. Door het systeem in meerdere iteraties te ontwikkelen, is er iedere twee weken een plan, demo en evaluatiemoment. Hierdoor is de kans dat het project slaagt groter, omdat de opdrachtgever nauw betrokken blijft en er continu ruimte is voor wijzingen en verbeteringen.

Iteratief werken kan ook door middel van andere methodieken. Er is echter gekozen voor de Agile Scrum methodiek, omdat deze genoeg elementen bevat die toegepast kunnen worden in dit project. Daarnaast hebben alle betrokkenen van het project ervaring met Agile Scrum.

4.2 Technieken

Tijdens het project zijn de onderstaande technieken gebruikt:

- **Visual Studio**
  Een softwarepakket voor het ontwikkelen van onder andere C# programmacode.
- **RStudio**
  Een softwarepakket voor het ontwikkelen van R programmacode. Tijdens de training "The Essentials of Programming in R" werd RStudio gebruikt en geadviseerd voor de ontwikkeling van R programmacode. Aangezien ik aan het begin van het project nog weinig ervaring had met de R programmeertaal heb ik er voor gekozen RStudio te blijven gebruiken.
- **Team Foundation Server (TFS)**
  Een softwarepakket voor onder andere versiebeheer van programmacode en ter ondersteuning van het scrum proces. Het hele scrumproces zoals de productbacklog, sprints en taken kunnen in TFS beheerd worden. De ontwikkelde progammacode kan in TFS gekoppeld worden aan taken die weer gekoppeld zijn aan product backlog items. Door deze integratie is uitgevoerd werk gekoppeld aan product backlog items.
- **Enterprise Architect**
  Een softwarepakket voor onder andere het ontwerpen van software en het vertalen van ontwerpen naar programmacode *(Sparx Systems, z.d.)*. Dit softwarepakket wordt door Info Support gebruikt en aangeboden.
- **Vagrant**
  Een softwarepakket waarmee een (virtuele) projectomgeving beschreven kan worden. Vervolgens kan met één handeling de beschreven projectomgeving gerealiseerd worden. Dit wordt bereikt door geautomatiseerde virtuele machines aan te maken en deze vervolgens te installeren, configureren en starten *(Vagrant, z.d.)*. Dit werd door experts van Info Support aangeraden.
- **RabbitMQ**
  Een softwarepakket dat gebruikt kan worden voor de communicatie tussen applicaties door middel van berichten *(RabbitMQ, z.d.)*. Er zijn meerdere softwarepakketten beschikbaar (zoals bijvoorbeeld ActiveMQ of Kafka) die deze functionaliteit bieden, maar RabbitMQ is gekozen omdat deze een duidelijke online documentatie inclusief voorbeelden hebben.
- **InfluxDB**
  Een softwarepakket waarmee tijd gerelateerde gegevens beheerd kunnen worden. Uit het onderzoek waarvan de resultaten te vinden zijn in bijlage *9.2 Bijlage B - Onderzoeksrapport* bleek dat InfluxDB een goede keuze is (binnen dit project) voor het beheren van tijd gerelateerde gegevens.

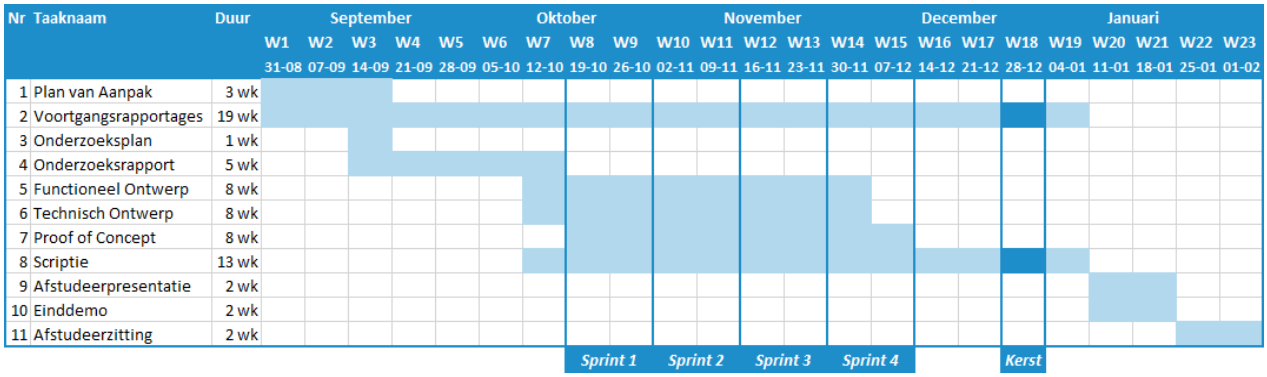
- **Redis**
  Een softwarepakket waarmee gegevens beheerd kunnen worden. Redis is naast InfluxDB gebruikt, omdat er naast tijd gerelateerde gegevens ook niet-relationele gegevens beheerd moesten worden. De keuze voor Redis is gemaakt, omdat het de gewenste functionaliteit biedt en in het onderzoek behandeld is.
- **ASP.NET Web API**
  Een softwareframework wat gebruikt kan worden voor de realisatie van webservices *(Microsoft Developer Network, z.d)*.
- **Microsoft Unity**
  Een softwareframework dat Dependency Injection mogelijk maakt *(Neill, 2011)*.
- **Log4Net**
  Een softwareframework dat diverse vormen van logging mogelijk maakt *(Apache Logging Services, z.d.)*.
- **R.Net**
  Een softwareframework dat een koppeling tussen de R - en C# programmeertaal mogelijk maakt *(CodePlex, z.d.)*. De keuze voor R.Net is genomen omdat deze de grootste online community heeft (op Github/Codeplex) met de meeste downloads op NuGet.
- **Moq**
  Een softwareframework waarmee delen programmacode vervangen kunnen worden bij het geautomatiseerd testen van programmacode. Hiermee kunnen bijvoorbeeld externe afhankelijkheden uitgesloten worden zodat dit de testen niet beïnvloedt.
- **Windows Management Instrumentation (WMI)**
  Een onderdeel van het Windows besturingssysteem waarmee systeeminformatie opgevraagd kan worden.

De technieken waar de keuze niet expliciet voor is beschreven, zijn keuzes die gemaakt zijn op basis van de ervaringen die de opdrachtnemer met deze technieken heeft.

4.3 Planning

4.3.1 Originele planning

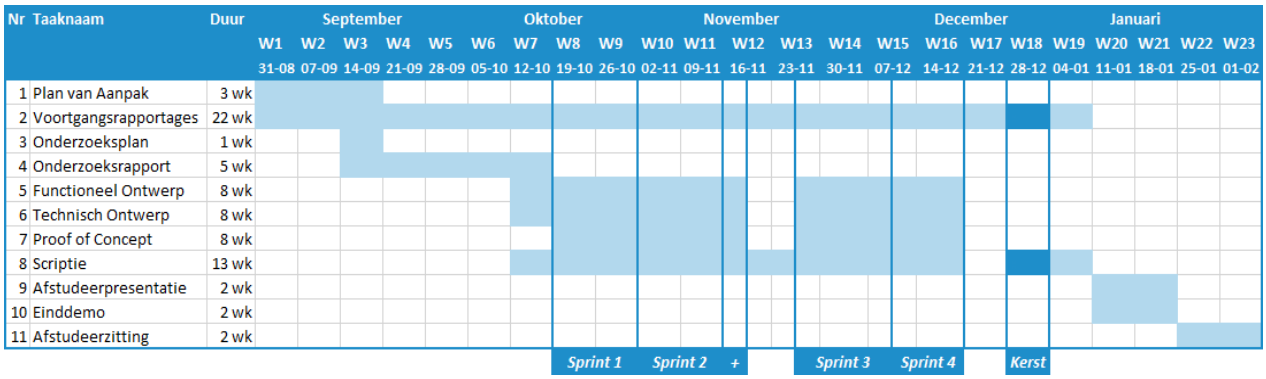
In *Figuur 2 - Gantt chart "Originele planning"* is de planning weergegeven zoals opgesteld in het Plan van Aanpak.



Figuur 2 - Gantt chart "Originele planning"

4.3.2 Afwijking op planning

Gedurende het project is er maar één keer afgeweken van de originele planning. Gedurende de tweede sprint is er door de opdrachtnemer en opdrachtgever besloten om de derde sprint met anderhalve week te verplaatsen. Dit is om meerdere redenen besloten, waarvan de belangrijkste reden het verkrijgen van test data was. Tussen de tweede en de derde sprint heeft er een korte sprint plaatsgevonden en is er een begin gemaakt met het schrijven van dit document. In de korte sprint is het gerealiseerde werk uitgebreid om test data te kunnen verkrijgen. In *Figuur 3 - Gantt chart "Uitgevoerde planning"* is de uitgevoerde planning weergegeven.



Figuur 3 - Gantt chart "Uitgevoerde planning"

5. Proces

5.1 Voortgangsrapportage

Gedurende het project is er eenmaal per twee weken een voortgangsrapportage verstuurd naar de opdrachtgever en alle begeleiders. In deze voortgangsrapportages werden in een paar pagina's de volgende onderdelen besproken:

- De uitgevoerde werkzaamheden;
- Eventuele afwijking op de planning;
- De planning van de volgende weken.

5.2 Plan van Aanpak

5.2.1 Uitvoering

Tijdens de eerste drie weken van dit project is een Plan van Aanpak opgesteld. In deze weken vonden er meerdere (kennismakings)gesprekken met alle betrokkenen plaats om de opdracht helder te krijgen. Daarnaast heb ik een driedaagse training gevolgd genaamd: "The Essentials of Programming in R." In deze training werd een introductie gegeven in de statistische programmeertaal, R. In deze introductie werd de syntax, veel gebruikte ontwikkelomgeving, het maken van analyses op datasets en het visualiseren van datasets en modellen behandeld. Daarnaast werd de theorie achter lineaire en logistische regressiemodellen behandeld. Dit zijn modellen die in R geprogrammeerd kunnen worden om voorspellingen van data te maken.

Na het volgen van de training en bij het helder krijgen van de opdracht realiseerde ik mij dat een goed vooronderzoek essentieel is om tot een bewezen concept te komen waarmee problemen voorspeld kunnen worden. Het opstellen van de deelvragen vormde een uitdaging en door het korte tijdsbestek was het moeilijk om een balans tussen de onderzoeks- en ontwikkelingsfase te vinden. De volgende deelvragen zijn daarom in samenspraak met de opdrachtgever geschrapt:

- Hoe zorg je ervoor dat de vertraging tussen analyseren en voorspellen zo minimaal mogelijk blijft?
- Hoe lang blijven gegevens relevant genoeg om bewaard te blijven?

Nadat het Plan van Aanpak opgesteld was en ik aan het wachten was op feedback van mijn begeleiders heb ik al een begin gemaakt aan de onderzoeksfase. Tijdens het schrijven van het onderzoeksplan zijn de onderzoeksvragen die opgenomen waren in het Plan van Aanpak nog aangescherpt.

5.2.2 Resultaten

Het Plan van Aanpak is op 28 september verstuurd naar Fontys Hogescholen dat vervolgens op 30 september is goedgekeurd door de assessor.

5.3 Onderzoeksfase

5.3.1 Uitvoering

In de eerste week van de onderzoeksfase is een onderzoeksplan geschreven. Tijdens het schrijven van het onderzoeksplan is begonnen met het oriënteren op de beschikbare literatuur en is de manier waarop het onderzoek uitgevoerd gaat worden vast gesteld.

In de daarop volgende weken is voornamelijk literatuuronderzoek uitgevoerd. Het geplande veldwerk heeft niet veel plaatsgevonden en is in de vorm van veelvoudige gesprekken met opdrachtgever en technisch begeleider uitgevoerd.

Eerst is onderzocht welke softwarepakketten er beschikbaar zijn voor het verkrijgen en centraliseren van (de inhoud van) logbestanden en systeeminformatie. Vervolgens is onderzocht welke vormen van centrale opslag mogelijk zijn. Hierbij is er naast de mogelijkheden ook een koppeling gemaakt naar de gevonden softwarepakketten voor het verkrijgen en centraliseren van gegevens. Daarna is onderzocht op welke manieren gegevens geanalyseerd kunnen worden om afwijkend gedrag te herkennen. Toen is onderzocht hoe door middel van afwijkend gedrag problemen herkent kunnen worden. Tot slot werd onderzocht hoe het herkennen van problemen kan leiden tot voorspellen.

De bevindingen werden wekelijks besproken met de opdrachtgever en technisch begeleider. Aan het einde van de onderzoeksfase is er een onderzoeksrapport geschreven met daarin alle bevindingen.

Het uitvoeren van een onderzoek was relatief nieuw en moeilijk voor mij. Daarnaast is het een complex project waarbij een breed vooronderzoek van belang is om tot een goed resultaat te kunnen komen. Daarom hebben gedurende de onderzoeksfase veel feedback momenten plaatsgevonden met de opdrachtgever en technisch begeleider. De focus bij het beantwoorden van deelvragen wist ik niet altijd goed te leggen en door bevindingen regelmatig terug te koppelen en feedback te vragen kon hierop indien nodig bijgestuurd worden. Door veel feedback te vragen heb ik wel veel geleerd over het uitvoeren van een onderzoek en hoe een onderzoeksrapport opgesteld moet worden.

5.3.2 Resultaten

Uit het onderzoek blijkt dat voor het verkrijgen en centraliseren van gegevens er verschillende softwarepakketten beschikbaar zijn. Hieruit ontstond een advies voor twee softwarepakketten. Deze softwarepakketten functioneren ongeveer hetzelfde en kunnen beiden (de inhoud van) logbestanden en systeeminformatie verzamelen en centraliseren.

Voor de centrale opslag van gegevens zijn verschillende oplossingen uitgewerkt waarbij naast een uitwerking van NoSQL oplossingen een andere vorm van centrale opslag aan het licht kwam genaamd “Time Series databasesystemen”. Deze variant databasesystemen zijn speciaal ontwikkeld voor de opslag en voor het analyseren van tijd gedreven gegevens.

Bij het beantwoorden van de deelvraag omtrent het herkennen van afwijkend gedrag is duidelijk geworden dat door patronen te ontdekken in gegevens het mogelijk wordt de toekomst te voorspellen. Met machine learning is het mogelijk te leren van gegevens. Dit resulteert in kennis die toegepast kan worden om verwacht gedrag te onderscheiden van afwijkend gedrag.

Gedurende het onderzoek bleek echter dat aan de hand van (de inhoud van) logbestanden het moeilijk is problemen te voorspellen. Daarnaast was al bekend bij het beantwoorden van de deelvraag omtrent het verkrijgen van gegevens, dat de applicaties die systeeminformatie aanbieden hun beperkingen hebben. Deze applicaties sluiten goed aan op Fluentd en Logstash, maar het interval waarmee systeeminformatie wordt aangeboden is echter te groot om realtime voorspellingen te kunnen maken. Met behulp van de klassen die het .NET framework beschikbaar stellen kan een applicatie ontwikkeld worden die met een kleinere interval eventueel meer of specifiekere informatie kan aanbieden.

Tijdens het onderzoeken van de laatste deelvraag werd duidelijk dat veel problemen gekoppeld kunnen worden aan verwacht afwijkend gedrag. Dat betekent dat door afwijkend gedrag te herkennen, problemen herkend en mogelijk voorspeld kunnen worden. Door middel van Time Series Analysis kan het ontstaan van afwijkend gedrag voor gegevens uit systeem-, beschikbaarheids- en reactie informatie voorspeld worden. Door de verwachting te vergelijken met het daadwerkelijke gedrag, wordt afwijkend gedrag bekend voordat problemen ontstaan. Hierdoor wordt het mogelijk problemen gekoppeld aan afwijkend gedrag uit dit soort gegevens te voorspellen voordat ze plaatsvinden.

Bovengenoemde resultaten vormen een beknopte samenvatting van enkele van de belangrijkste bevindingen. De volledige bevindingen en een allesomvattende conclusie en aanbevelingen van het onderzoek zijn beschreven in het onderzoeksrapport, te vinden in bijlage *9.2 Bijlage B - Onderzoeksrapport*.

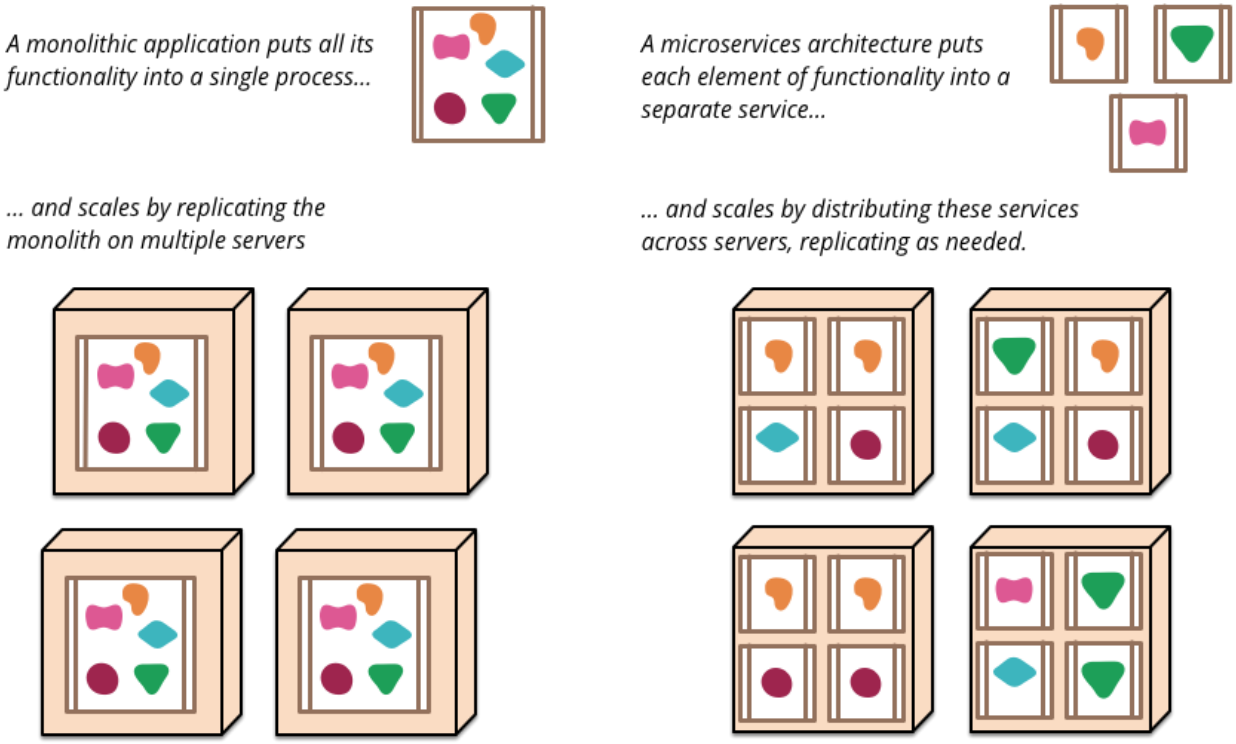
5.4 Ontwikkelingsfase

5.4.1 Uitvoering

De ontwikkelingsfase is in meerdere iteraties uitgevoerd en voor iedere iteratie was er een helder doel. Voor het starten van een iteratie werden de eisen en wensen van de opdrachtgever nogmaals gecontroleerd en aangescherpt voordat de sprint gestart werd. Bij het starten van een sprint werden de eisen en wensen van die sprint vertaald in een functioneel en technisch ontwerp. Deze ontwerpen resulteerden in een (bijgewerkt) functioneel (zie bijlage *9.3 Bijlage C - Functioneel Ontwerp*) en technisch ontwerp (zie bijlage *9.4 Bijlage D - Technisch Ontwerp*) dat vervolgens gebruikt werd om de eisen en wensen te realiseren. Aan het einde van iedere sprint is het resultaat in de vorm van een demonstratie getoond aan de opdrachtgever.

5.4.1.1 Sprint 1

Het doel van de eerste sprint was het verkrijgen en bewaren van informatie. Omdat het een proof of concept is voor een systeem dat veel informatie moet verwerken is er gekozen voor een Microservices architectuur. Dit is een architectuur waarbij het systeem in afzonderlijke componenten opgebouwd wordt in plaats van in één grote applicatie (ook wel monoliet) zie *Figuur 4 - Microservices architectuur* .



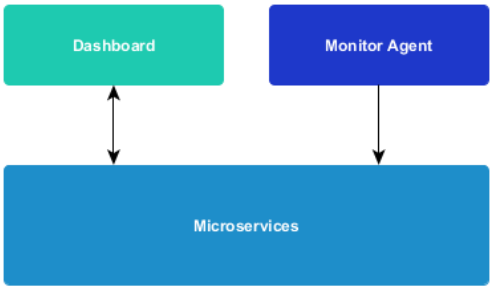
Figuur 4 - Microservices architectuur (Fowler & Lewis, 2014)

Het gebruik van een Microservices architectuur biedt een aantal voordelen, waaronder de mogelijkheid om voor ieder component aparte keuzes te kunnen maken. Doordat ieder component op zichzelf staat, kan voor iedere microservice de beste implementatie en dataopslag keuze gemaakt worden, zonder dat er rekening gehouden hoeft te worden met de rest. Door de flexibelere keuze ontstaat er echter wel een hogere totale complexiteit, maar is de complexiteit per component wel nog altijd veel lager dan bij een grote applicatie.

Een ander groot voordeel is de flexibiliteit en schaalbaarheid van het systeem. Bij een systeem dat bestaat uit één grote applicatie, kan er alleen geschaald worden door de hele applicatie meerdere keren uit te voeren. Bij de Microservices architectuur is het mogelijk om specifiek de intensieve componenten vaker uit te voeren om de totale prestatie van het systeem te verbeteren.

Naast de Microservices architectuur is een andere belangrijke keuze het zelf ontwikkelen van een applicatie voor het verkrijgen van systeeminformatie en logbestanden, in plaats van het gebruik maken van bestaande softwarepakketten. Uit het onderzoek werd duidelijk dat aan de hand van systeeminformatie het beste voorspellingen gemaakt kunnen worden. Er zijn softwarepakketten die hier iets mee kunnen, maar hier zitten beperkingen aan.

Het verkrijgen van systeeminformatie en logbestanden moet ten alle tijde gebeuren en daarom is er besloten een Windows Service applicatie te ontwikkelen. Dit type applicatie kan door Windows op de achtergrond worden uitgevoerd en wordt automatisch gestart bij het starten van Windows. De benodigde systeeminformatie wordt met behulp van het .NET framework opgevraagd uit WMI.



Figuur 5 - Abstracte architectuur

Door de keuze voor een Microservices architectuur ontstond de abstracte architectuur die weergegeven is in *Figuur 5 - Abstracte architectuur*.

Het component "Monitor Agent" hierin is de Windows Service applicatie die met een interval verkregen informatie verstuurd naar het (centrale) systeem.

Via het component "Dashboard" kunnen gebruikers verkregen informatie en voorspellingen bekijken.

5.4.1.2 Sprint 2

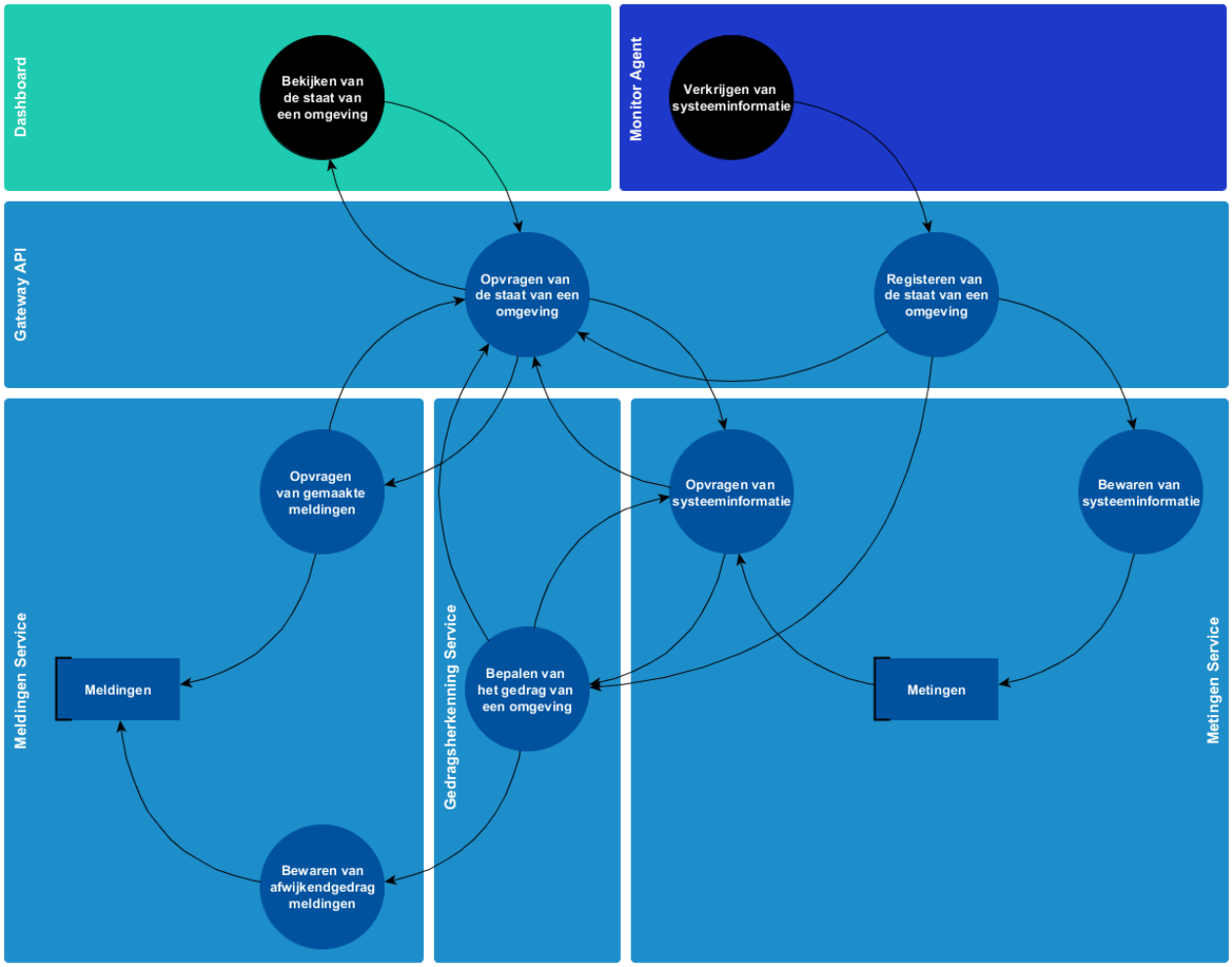
Het doel van de tweede sprint was een begin maken met het bepalen van het gedrag van verkregen systeeminformatie en het tonen van gemaakte meldingen in een dashboard. Bij het bepalen van het gedrag moet er aan de hand van systeeminformatie uit het verleden bepaald worden of nieuwe metingen afwijkend gedrag vertonen.

Door dit te verwezenlijken, wordt de complexiteit van het systeem een stuk hoger. Naast het doorgeven of opvragen van informatie tussen enkele componenten wordt er nu door meerdere verschillende componenten tegelijk op dezelfde informatie gereageerd (zie *Figuur 6 - Processen en componenten*). Bij het registeren van de staat van een omgeving worden de onderstaande processen naast elkaar uitgevoerd:

- De verkregen systeeminformatie wordt bewaard door het component "Metingen service".
- Aan de hand van de verkregen systeeminformatie wordt het nieuwe gedrag van een omgeving bepaald door het component "Gedragsherkenning service".
- De verkregen systeeminformatie wordt doorgegeven aan het component "Dashboard".

Hierbij kan het zijn dat bij het bepalen van het gedrag van een omgeving het verleden van deze omgeving onbekend is of dat het voorspelmodel niet meer recent is. Wanneer dit het geval is moeten metingen uit het verleden opgevraagd worden uit het component "Metingen service".

Door deze extra complexiteit blijkt het realiseren van processen die naast elkaar plaatsvinden, moeilijker te zijn dan verwacht. Niet alleen de communicatie tussen meerdere componenten, maar ook het tegelijkertijd afhandelen van verzoeken binnen hetzelfde component kan leiden tot problemen. Ondanks dat er testen zijn geschreven voor alle lagen binnen alle componenten, is het vaak raadzaam de integratie tussen deze lagen en componenten ook te testen. Hoewel deze complexiteit verkeerd ingeschat is, zijn de afgesproken functionaliteiten voor sprint 2 binnen de afgesproken tijd gerealiseerd. Er is namelijk bij het plannen van sprint 2 extra tijd gereserveerd voor het ontwerpen en realiseren van het component "Gedragsherkenning service".



Figuur 6 - Processen en componenten

Om het dashboard zo actueel mogelijk te houden is realtime informatieverstrekking gewenst. Bij “traditioneel” berichtenverkeer tussen webapplicaties en webservices is het echter alleen maar mogelijk vanaf de webapplicaties aanvragen in te dienen bij de webservices. Wanneer realtime informatieverstrekking gewenst is, moet de webapplicatie bijvoorbeeld iedere seconde een aanvraag indienen bij een webservice. Om dit op te lossen is er gekozen voor web sockets. Met web sockets is naast berichtenverkeer van web applicatie naar webservice ook berichtenverkeer van web service naar webapplicatie mogelijk.

Wanneer het component “Dashboard” de staat van een omgeving opvraagt bij het component “Gateway API”, wordt tussen beide componenten een websocket verbinding tot stand gebracht. Vervolgens wordt naast het opvragen van eerder gemaakte meldingen en systeeminformatie geabonneerd op het ontvangen van nieuwe meldingen en systeeminformatie verzoeken. Door gebruik te maken van websockets en het abonneren op nieuwe meldingen en systeeminformatie verzoeken, wordt het mogelijk nieuwe meldingen en systeeminformatie door te sturen van het component “Gateway API” naar het component “Dashboard”. Hierdoor blijft het component “Dashboard” actueel.

5.4.1.3 Korte sprint

Zoals in hoofdstuk *4.3.2 Afwijking op planning* te lezen was, is een korte sprint geïntroduceerd met als voornamelijk doel het verkrijgen van test data. In principe kon de Windows Service daarvoor direct gebruikt worden, maar omdat ik verwachtte dat deze nog niet compleet genoeg was, heb ik met de opdrachtgever een brainstormsessie gehouden. Tijdens deze sessie werden verschillende ideeën besproken waaronder mijn idee om het geheugen- en processorverbruik voor ieder uitgevoerd proces te meten. Vervolgens is er door de opdrachtgever besloten de Windows Service uit te breiden met het meten van het geheugen en processorverbruik voor ieder uitgevoerd proces, met uitzondering van processen die door het besturingssysteem zelf uitgevoerd worden.

5.4.1.4 Sprint 3

Het doel van de derde sprint was het verder uitwerken en bruikbaar maken het systeem. Het verkrijgen, bewaren en inzien van gegevens voor individuele machines was goed genoeg maar er was behoefte aan een overzicht waarin de globale actuele staat van meerdere machines inzichtelijk wordt. Daarnaast moest de verkregen data geanalyseerd worden en moest het component “Gedragsherkenning service” verder uitgewerkt worden zodat deze voorspellingen kan maken.

Het realiseren van het gewenste overzicht was niet moeilijk omdat de benodigde architectuur om dit te realiseren al aanwezig was. Het analyseren van de data was echter wel heel moeilijk en kreeg nog een onverwachtse wending. De belangrijkste gemaakte keuzes en leermomenten van deze sprint zaten daarom voornamelijk bij het analyseren van de data.

Bij het analyseren van de data werd al snel duidelijk dat we dat we allemaal iets over het hoofd hadden gezien en/of verkeerde aannames hebben gemaakt. Om afwijkend gedrag te kunnen voorspellen moeten fouten uit het verleden bekend zijn. Wanneer fouten uit het verleden bekend zijn kan geanalyseerd worden of en hoe afwijkend gedrag uit fouten herleid kan worden. De verzamelde gegevens bevatten veel observaties maar geen classificatie of een fout heeft plaatsgevonden tijdens een observatie. Het handmatig classificeren van gegevens aan de hand van bepaalde criteria (bijvoorbeeld het verbruik is hoger dan x) bleek geen bruikbare resultaten te bieden.

In overleg met de opdrachtgever is besloten de verkregen data te verrijken met geregistreerde foutmeldingen. Aangezien het op voorhand niet bekend was of het verrijken van de verkregen data het gewenste resultaat zou bieden heb ik besloten hiervoor een zo minimalistisch mogelijke applicatie te realiseren buiten het systeem. De gerealiseerde applicatie kon verschillende soorten logbestanden inlezen en koppelen aan de verkregen data. Ook uit de verreikte data kwamen geen bruikbare resultaten.

Aangezien de tijd van het project beperkt was en ik dit probleem niet meer alleen kon oplossen, heb ik aan de bel getrokken bij de opdrachtgever om samen een oplossing te bedenken. Bij het zoeken naar een oplossing werd als eerste gekeken of we binnen een korte tijd geclassificeerde data beschikbaar konden krijgen. Er waren echter op dat moment geen omgevingen beschikbaar die afwijkend gedrag of foutief gedrag vertonen. Het installeren van mijn of andere tools op een omgeving had dus weinig zin, omdat het best mogelijk is dat voor het einde van het project er nog steeds geen fout heeft plaats gevonden. Daarnaast was er maar een beperkte hoeveelheid data beschikbaar van omgevingen waarop in het verleden fouten hadden plaats gevonden.

Het verkrijgen van betere data was niet mogelijk en we besloten daarom samen om een andere richting in te slaan. De opdrachtgever gaf aan te willen weten of het mogelijk was pieken in verbruik te voorspellen. Ik heb daarop voorgesteld niet alleen de pieken maar het (totale) processor verbruik van omgevingen te voorspellen. Tijdens het onderzoek heb ik wetenschappelijke artikelen gelezen over het kort vooruit voorspellen van computer verbruik.

De belangrijkste bevindingen bij het analyseren van de data heb ik verzameld en beschreven in de vorm van een data-analyse, te vinden in bijlage *9.6 Bijlage F - Data-analyse*.

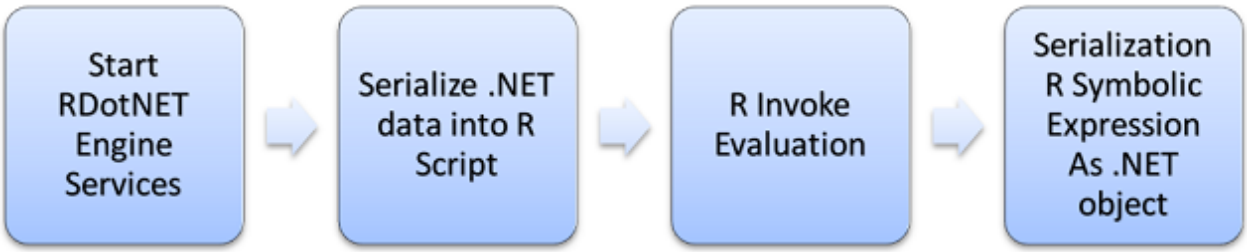
Het was een rommelige sprint waarin ik aangetoond heb actief te kunnen handelen op (grote, onverwachtse) problemen die gedurende een sprint kunnen ontstaan. Naast het zelfstandig zoeken naar oplossingen heb ik in mijn ogen op de juiste momenten de opdrachtgever betrokken waardoor ik weinig tijd verloren heb.

5.4.1.5 Sprint 4

Het voornaamste doel van de vierde sprint was het onderzoeken hoe het (totale) processor verbruik voorspeld kan worden. Indien dit mogelijk was moest het huidige proof of concept systeem met deze vorm van voorspellen uitgebreid worden.

Het voorspellen van het (totale) processor verbruik bleek mogelijk te zijn door middel van het ETS model. Wanneer dit model wordt getraind met 2 uur aan processor informatie uit het verleden, wordt het voorspellen van het (totale) processor verbruik in de toekomst mogelijk. Bij voorspellingen 10 seconden de toekomst in wordt een nauwkeurigheid van ongeveer 90% (bij een absolute afwijking van maximaal 3%) behaald. Hoe tot dit model is gekomen en een uitgebreidere beschrijving van dit model zijn te vinden in *hoofdstuk 3 Forecasting* van de Data-analyse, te vinden in bijlage *9.6 Bijlage F - Data-analyse*.

Een van de uitgangspunten die in het Plan van Aanpak staat beschreven is: “Analyses en voorspellingen worden gemaakt in de programmeertaal R (ontwikkeld voor statistische doeleinden).” Naast dit uitgangspunt is het niet handig het wiel opnieuw uit te willen vinden in een andere programmeertaal. Er is daarom gekozen voor een hybride oplossing waarbij via .NET programmatuur R programmacode wordt uitgevoerd (zie *Figuur 7 - R integratie .NET*). In de programmeertaal R wordt het model getraind en worden voorspellingen gemaakt. Vervolgens worden de gemaakte voorspellingen uit R omgezet naar objecten die in .NET gebruikt kunnen worden. De hybride oplossing is gerealiseerd met behulp van het R.NET softwareframework.



Figuur 7 - R integratie .NET (Guigang, 2015)

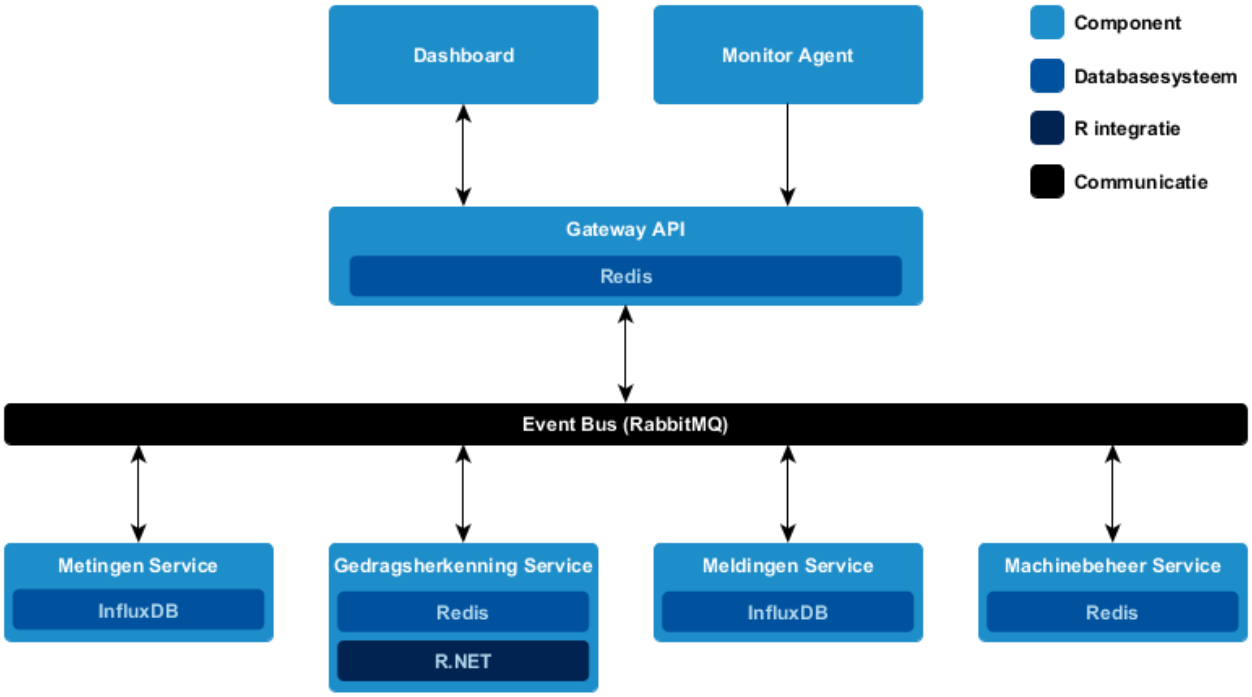
Gedurende de sprint ben ik erachter gekomen dat de statistische berekeningen zoals het trainen van een model of het maken van een voorspelling veel systeemkracht vereist. In mijn ontwikkelomgeving (zie bijlage *9.7 Bijlage F – Specs Ontwikkelomgeving*) duurde het trainen van één model (met 3 uur aan data) al enkele seconden. Wanneer er met meerdere modellen naast elkaar of met meer data wordt gewerkt kan dit oplopen van enkele minuten tot zelfs uren.

Bij het starten van de sprint was het nog niet bekend of het voorspellen van het (totale) processor verbruik mogelijk is en hoe de implementatie hiervan gaat uitzien. Hierdoor is het lastig tot onmogelijk ontwerpen vooraf te maken. De opdrachtgever had besloten dat voor deze sprint het functioneel en technisch ontwerp niet verder uitgewerkt hoeft te worden.

5.4.2 Resultaten

Het maken van voorspellingen is een proces dat bestaat uit een aantal stappen. Als eerste moeten de benodigde gegevens verkregen worden. Deze gegevens moeten vervolgens voor een x tijd bewaard worden en oudere gegevens die niet relevant meer zijn moeten automatische opgeruimd worden zodat het systeem niet eindeloos blijft groeien. Daarna moeten de bewaarde gegevens geanalyseerd worden om voorspellingen te kunnen maken. Tot slot moeten de gemaakte voorspellingen inzichtelijk zijn.

Het opgeleverde systeem heeft bewezen dat het technisch mogelijk is alle stappen (realtime) te doorlopen en samen te brengen in een systeem. De uiteindelijke architectuur van het opgeleverde systeem is te zien in *Figuur 8 - Systeem architectuur*.



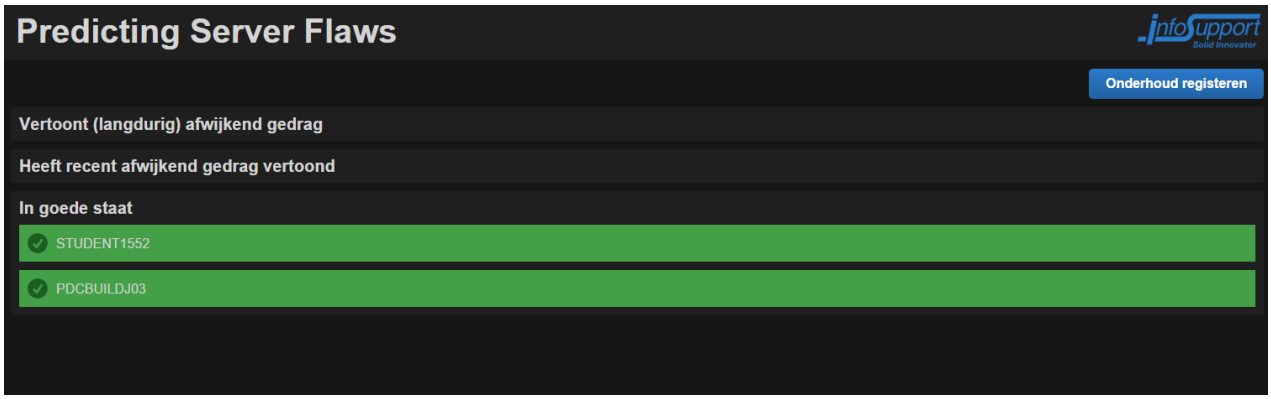
Figuur 8 - Systeem architectuur

Door middel van meerdere "Monitor Agent" componenten worden de benodigde gegevens met een interval verkregen en verstuurd naar het (centrale) systeem. Het "Gateway API" component ontvangt deze gegevens via een (beveiligd) gestandaardiseerd communicatieprotocol (HTTP(S)). De ontvangen gegevens worden vervolgens bekend gemaakt aan de rest van het (centrale) systeem.

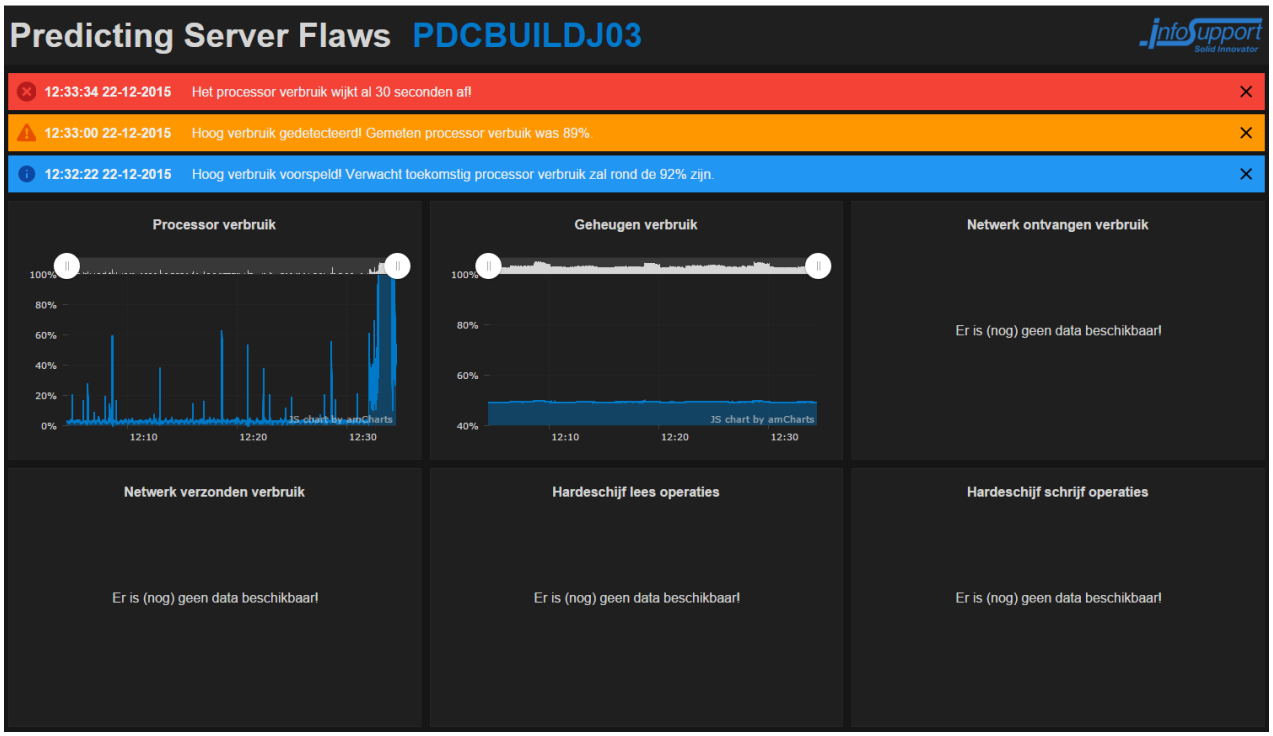
Deze bekend gemaakte gegevens worden vervolgens bewaard door het "Metingen Service" component en tegelijkertijd geanalyseerd door het "Gedragsherkenning Service" component. Bij het analyseren van de bekend gemaakte gegevens wordt eventueel hoog verbruik gedetecteerd en wordt een voorspelling gemaakt aan de hand van gegevens uit het verleden. Indien de gegevens (sterk) afwijken van de verwachting of wanneer er hoog verbruik in de toekomst verwacht wordt, zal dit gemeld worden aan het (centrale) systeem.

Het component "Meldingen Service" bewaart gemaakte meldingen zodat gemaakte meldingen ook later opnieuw opgevraagd kunnen worden. Het component "Machinebeheer Service" beheert welke machines er bekend zijn en wat hun actuele statussen zijn. Wanneer er (te) veel meldingen worden gemaakt over een machine in een korte tijd, verandert de status van een machine bijvoorbeeld van "gezonde staat" in "slechte staat".

Via het component “Dashboard” kunnen gebruikers (realtime) een overzicht krijgen van alle machines en hun actuele staat (zie *Figuur 9 - Overzicht scherm dashboard*). Daarnaast kunnen ze (realtime) per machine bekijken welke informatie er verkregen is en welke voorspellingen en/of meldingen er gemaakt zijn (zie *Figuur 10 - Machine detailscherm dashboard*).



Figuur 9 - Overzicht scherm dashboard



Figuur 10 - Machine detailscherm dashboard

5.5 Overige activiteiten

Gedurende de afstudeerstage zijn er tijdens werkuren meerdere presententaties en trainingen gevolgd. Daarnaast zijn er buiten werktijd ook meerdere Info Support Kennis Avonden (ISKA's) gevolgd. Tijdens een ISKA geven collega's presentaties over nieuwe technieken of opgedane ervaringen bij projecten. De gevolgde activiteiten zijn opgenomen in *Tabel 3 - Overige activiteiten*.

Datum	Type	Titel	Gegeven door
01-09-2015	Presentatie	Plan van Aanpak	Gert Jan Timmerman
02-09-2015	ISKA	Info Support Datacenter Architectuur	Willem Jan Vastenholt, Roland Noordermeer en Rob te Boekhorst
02-09-2015 t/m 04-09-2015	Training	The Essentials of Programming in R	Maarten van Duuren
16-09-2015	Kwartaaloverleg	Strategie 2015 Accelerate the business, Foresight acting	Norbert Fassotte, Guido Derkx en Dennis Joosten
17-09-2015	Presentatie	Hoe zet ik een onderzoek op?	Gert Jan Timmerman
29-09-2015	Presentatie	Functioneel en technische ontwerp	Gert Jan Timmerman
30-09-2015	ISKA	Het bouwen van een nieuw peer-to-peer betaalplatform bij de ING	Sander de Groot
05-10-2015	Training	Training projectplanning en documentatie voor afstudeerders	Remi Kok
07-10-2015	ISKA	De Bedrijfsatleet	Koen Gonnissen
28-10-2015	Presentatie	Het schrijven van het eindverslag	Gert Jan Timmerman
28-10-2015	ISKA	Performance testing van Microservices	Rolf Huisman
18-11-2015	Kwartaaloverleg	Strategie 2015 Accelerate the business, Innovatieve technologie	Henk Brands, Pim Tegelaar en Remy Hansen
02-12-2015	Presentatie	Beyond bullet points	Harry Nieboer
16-12-2015	ISKA	Using SpecFlow, the .Net Specification by Example engine	Mark Taling

Tabel 3 - Overige activiteiten

6. Conclusie en aanbevelingen

6.1 Conclusie

De doelstelling van de opdrachtgever zoals beschreven in hoofdstuk *3.3 Doelstelling* is niet helemaal behaald. Het voorspellen van problemen is namelijk niet gelukt. Wel wordt het middels het (proof of concept) systeem mogelijk proactief te handelen. Het resultaat is ook anders dan beschreven in hoofdstuk *3.4 Resultaat*.

Gedurende de uitvoering van de ontwikkelfase bleek namelijk dat er aan de hand van de verkregen gegevens geen bruikbare voorspellingen konden worden gemaakt over wanneer problemen en/of afwijkend gedrag gaan ontstaan. Om de doelstelling tot proactief handelen te verwezenlijken is daarom besloten verbruiken te voorspellen. Het voorspellen van verbruiken kan namelijk resulteren in het voorspellen van afwijkend gedrag dat ontstaat door hoge of afwijkende verbruiken. Er kan namelijk worden voorspeld dat er voor een x periode een y verbruik gaat plaatsvinden. Of en hoe afwijkend gedrag gerelateerd is aan één of meerdere verbruiken is nu echter nog onbekend.

Door de toepassing van Agile Scrum ontwikkelmethodiek is het echter niet erg dat het eindresultaat afwijkt van het resultaat beschreven aan het begin van het project. Gedurende de ontwikkelfase is er namelijk voor iedere sprint in samenspraak met de opdrachtgever een resultaat bepaald. De afgestemde resultaten werden aan het einde van iedere sprint behaald.

Het opgeleverde systeem bewees dat het mogelijk was realtime gegevens te verkrijgen (en bewaren), te analyseren en daar vervolgens realtime voorspellingen mee te maken. De verkregen gegevens en gemaakte voorspellingen zijn inzichtelijk in een dashboard. Het systeem kan direct ingezet worden en kan waar nodig uitgebreid worden.

De integratie die in dit systeem gerealiseerd en bewezen is, met name de integratie tussen de statistische programmeertaal R en .NET, was waardevolle kennis voor het bedrijf. Hierover is een artikel gepubliceerd op KnowNow, het kennisdelingsplatform van Info Support. Zie bijlage *9.8 Bijlage G – Artikel KnowNow* voor het volledige artikel.

Voorafgaande aan de realisatie van het systeem is er een onderzoek uitgevoerd waarin het verkrijgen, bewaren en analyseren van gegevens wordt behandeld. Bij het analyseren van gegevens is er onderzocht hoe afwijkend gedrag herkend zou kunnen worden en hoe voorspellingen gemaakt kunnen worden. Dit onderzoek resulteerde in theoretische kennis die noodzakelijk was om aan de ontwikkeling van een systeem te kunnen beginnen.

6.2 Aanbeveling

Uit de resultaten van het onderzoek en de ontwikkeling van het (proof of concept) systeem is het duidelijk dat er technisch en theoretisch veel mogelijk is, maar dat er in de praktijk ook de juiste gegevens voor beschikbaar moeten zijn. Er wordt daarom aanbevolen een langdurige tijd (bijvoorbeeld een jaar) verschillende soorten gegevens te verzamelen en te bewaren. De momenten waarop fouten of afwijkend gedrag worden geconstateerd moeten gemarkeerd worden in de verzamelde gegevens zodat geclassificeerde gegevens beschikbaar worden. Het beschikbaar hebben van de juiste gegevens was in dit project de grootste uitdaging. Gedurende een project is er vaak te weinig of geen tijd meer om aan (de juiste) gegevens te komen.

Technisch en theoretisch is veel mogelijk en er wordt daarom aanbevolen een vervolgonderzoek en/of project te starten wanneer geclassificeerde gegevens beschikbaar zijn. Het systeem dat in dit project is opgeleverd is direct bruikbaar en kan waar nodig uitgebreid worden om meer of andere gegevens te verkrijgen. Ook de integratie met de statistische programmeertaal R functioneert naar behoren en kan uitgebreid of aangepast worden wanneer er een ander/beter of meerdere (voorspel)modellen toegepast moeten worden.

De technieken die in dit project zijn toegepast hebben bewezen dat het technisch mogelijk is realtime gegevens te analyseren om daar vervolgens realtime voorspellingen mee te maken. Het opdelen in services heeft zich vooral bewezen omdat de statistische berekeningen veel systeemkracht vereisen en daarom het beste meervoudig uitgevoerd moeten worden (om realtime te kunnen blijven functioneren).

7. Reflectie

Tijdens mijn afstudeerperiode heb ik veel nieuwe (technische) vakinhoudelijke en (algemene) kennis opgedaan. Dit begon al in de eerste week met een training die mij een introductie gaf in de statistische programmeertaal R. Naast de syntax was er ook een algemene introductie in de statistiek, zodat je de toepassing van R in de praktijk kon zien. Kennis van statistiek had ik nog niet en wiskunde heb ik sinds de middelbare school niet meer gehad.

Bij de start van dit project en het uitvoeren van mijn onderzoek heb ik veel nieuwe (technische) kennis opgedaan. Tijdens de uitvoering van mijn onderzoek ben ik in aanraking gekomen met populaire termen zoals NoSQL en Machine Learning. Bij het onderzoeken van de term NoSQL en de daar bijhorende (populaire) databasesystemen ben ik er achter gekomen dat er nog meer varianten van databasesystemen bestaan. Ik ontdekte bijvoorbeeld Time Series databasesystemen. Bij het onderzoeken van Machine Learning kwam ik erachter dat Machine Learning meer een aanpak beschrijft met daarin oplossingsrichtingen dan de oplossing zelf. Het oplossen van vraagstukken gebeurt aan de hand van (wiskundige) algoritmes die veelal tientallen tot zelfs honderden jaren bestaan. Door te experimenteren met verschillende algoritmes wordt pas duidelijk welke algoritmes in bepaalde situaties het beste werken. Dit is dus geen exacte wetenschap, hoewel ik dit in het begin wel dacht.

Door de uitvoering van het onderzoek heb ik veel ervaring opgedaan over hoe een onderzoek uitgevoerd moet worden. Dit was aan het begin van mijn afstudeerperiode nog relatief nieuw en moeilijk voor mij. Het behouden van de juiste focus in een onderzoek vond ik het moeilijkst en hiervan heb ik het meest geleerd. Bij het onderzoeken van algoritmes vond ik het bijvoorbeeld moeilijk de juiste focus te behouden. Het was hierbij van belang dat ik globaal wist welke varianten er beschikbaar waren, maar niet de exacte werking van ieder algoritme. Vanuit mijn karakter wil ik altijd graag alles van A tot Z weten. Dit is echter niet altijd noodzakelijk of van belang. Daarnaast wist ik tijdens mijn onderzoek natuurlijk nog niet welke algoritmes ik zou gaan gebruiken.

Omdat het een complex project was waarbij een breed vooronderzoek essentieel was, hebben er gedurende de onderzoeksfase veel feedbackmomenten plaatsgevonden met de opdrachtgever en technisch begeleider. Door bevindingen regelmatig terug te koppelen en feedback te vragen heb ik veel geleerd over het uitvoeren van een onderzoek en hoe een onderzoeksrapport opgesteld moet worden.

Tijdens de uitvoering van mijn ontwikkelfase heb ik meer ervaring opgedaan in het ontwerpen van software. Hoewel dit behandeld was op school, had ik er nog weinig ervaring mee in de praktijk. Ook bij het toepassen van de Agile Scrum ontwikkelmethodiek is het nog altijd van belang om ontwerpen te blijven maken. Dit zijn niet direct volledige ontwerpen, maar wel groeiende ontwerpen die na iedere iteratie meer vorm krijgen.

Het toepassen van nieuwe technieken was heel leerzaam. Voordat ik aan dit project begon, heb ik veel gebruik gemaakt van bestaande of nieuwe varianten van technieken. Bij het toepassen van nieuwe technieken kwam ik er achter dat nieuwe technieken niet altijd zo volledig zijn als ik gewend ben. InfluxDB, het Time Series databasesysteem dat ik behandeld heb in mijn onderzoek, heeft bijvoorbeeld geen functionerende .NET koppeling. De beschikbare koppelingen waren bijvoorbeeld niet actueel (omdat de techniek zich te snel vernieuwd) of werkten gewoonweg niet na behoren. Waar ik normaal direct aan de slag kon na het downloaden van een koppeling (stukje softwarecode), moest ik nu zelf een koppeling realiseren. Dit was achteraf niet moeilijk, omdat er een duidelijke documentatie beschikbaar is. Maar het is wel anders werken dan ik gewend ben. Ik kan me goed voorstellen dat dit voor veel mensen zorgt voor een drempel om een techniek niet te gaan gebruiken.

Hoewel ik al (een halfjaar) ervaring heb opgedaan met een Service-oriented architectuur, wat veel overeenkomsten heeft met een Microservices architectuur, heb ik toch een aantal (kleine) problemen gehad waarvan ik geleerd heb. Enkele voorbeelden van die problemen zijn de asynchrone communicatie tussen meerdere componenten en het tegelijkertijd afhandelen van meerdere verzoeken binnen hetzelfde component. Ik heb achteraf geleerd dat het soms ook handig is integratietesten te schrijven waarbij het geheel getest wordt. Wanneer iedere laag afzonderlijk wordt getest, dek je niet de communicatie tussen componenten of het gedrag wanneer meerdere instanties het zelfde proberen te doen.

Het belangrijkste dat ik tijdens mijn afstudeerperiode heb geleerd is het omgaan met onvoorziene omstandigheden. Na het analyseren van de verkregen data bleek dat deze data niet bruikbaar genoeg was om voorspellingen mee te maken. Dit was voor mij moeilijk om te accepteren, omdat ik overal altijd het beste van wil maken en een positief resultaat had verwacht te bereiken. Projecten die ik op school of bij mijn bijbaan (als applicatieontwikkelaar) heb uitgevoerd gingen altijd goed omdat daar natuurlijk van te voren al van bekend was dat het mogelijk was. Tijdens mijn afstudeerstage was dit nog niet bekend, anders hoefde dit project ook niet uitgevoerd te worden.

Het feit dat het in de gegeven situatie niet mogelijk was voorspellingen te maken, is echter ook een geldig en waardevol resultaat. Hoewel ik had bewezen dat het voorspellen van fouten niet mogelijk was ben ik toch verder gaan zoeken naar voorspellingen die wel mogelijk zijn. Bij het zoeken naar oplossingen heb ik veel (statistische) kennis opgedaan over het analyseren van data. Bij het analyseren van data heb ik veel geëxperimenteerd met verschillende algoritmes. Om de resultaten van experimenten te kunnen begrijpen en vergelijken was het noodzakelijk dat ik de globale verschillen tussen algoritmes begreep.

Het uitwisselen van ideeën over hoe de data geanalyseerd zou kunnen worden is in overleg met mijn technisch begeleider en opdrachtgever gebeurd. De voorgelegde ideeën en de uitwerkingen van de gekozen ideeën is echter zelfstandig uitgevoerd.

Tot slot hebben de Info Support Kennisavonden (ISKA's) en kwartaal overleggen mij een beter beeld gegeven van het bedrijfsleven. Tijdens de ISKA's werden nieuwe technieken en/of opgedane ervaringen bij projecten besproken. Hierin werd niet alleen verteld wat er goed ging, maar ook wat fout ging. Veel besproken fouten waren bijvoorbeeld de valkuilen van het toepassen van bepaalde nieuwe technieken. Op deze avonden werden voornamelijk onderwerpen behandeld waarvan ik nog geen of beperkte kennis had.

In de toekomst wil ik mij graag verder verdiepen in de brug tussen de software- en statistische wereld. Er is steeds meer data beschikbaar en ik verwacht dat er een grotere vraag zal zijn om meer met bestaande data te gaan doen. Naast voorspellingen kunnen er bijvoorbeeld ook aanbevelingen gemaakt worden. Ik vind de ontwikkelingen van software nu geweldig maar ik denk niet dat ik mijn hele leven alleen bezig wil zijn met programmeren. De extra dimensie die het analyseren van gegevens met zich meebrengt, spreekt me erg aan. De ambitie om in de toekomst door te stromen richting bijvoorbeeld het beroep "Data Scientist" is daarom aanwezig.

8. Literatuurlijst

Apache Logging Services. (z.d.). *Apache log4net.* Opgeroepen op november 20, 2015, van Apache Logging Services: https://logging.apache.org/log4net/

Christensson, P. (2010, april 14). *Log File Definition.* Opgeroepen op september 18, 2015, van Techterms: http://techterms.com/definition/logfile

Clifton, M. (2003, november 4). *What Is A Framework?* Opgeroepen op november 23, 2015, van Code Project: http://www.codeproject.com/Articles/5381/What-Is-A-Framework

CodePlex. (z.d.). *R.NET - Documentatie.* Opgeroepen op december 24, 2015, van CodePlex: http://rdotnet.codeplex.com/documentation

Fowler, M., & Lewis, J. (2014, maart 25). *Microservices.* Opgeroepen op november 20, 2015, van Martin Fowler: http://martinfowler.com/articles/microservices.html

Guigang, X. (2015, maart 25). *R language S4Object Serialization to .NET Object.* Opgeroepen op december 24, 2015, van Code Project: http://www.codeproject.com/Articles/890099/R-language-S-Object-Serialization-to-NET-Object

Info Support. (z.d.). *Over Info Support.* Opgeroepen op september 14, 2015, van http://www.infosupport.com/info-support-b-v/

Kernerman Dictionaries. (z.d.). *Algoritme Nederlands woordenboek.* Opgeroepen op oktober 2, 2015, van Kernerman Dictionaries: http://www.woorden.org/woord/algoritme

Khan Zaki, A. (2014, mei). *NoSQL databases: New millennium database for big data.* Opgeroepen op september 25, 2015, van http://esatjournals.org/Volumes/IJRET/2014V03/I15/IJRET20140315080.pdf

Marqit. (z.d.). *Marqit.* Opgeroepen op januari 4, 2016, van Business Intelligence: http://www.marqit.nl/wat-is-business-intelligence

Microsoft Developer Network. (z.d). *ASP.NET Web API.* Opgeroepen op november 20, 2015, van Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

Munoz, A. (z.d.). *Machine Learning and Optimization.* Opgeroepen op oktober 2, 2015, van https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf

Neill, B. (2011, september 19). Opgeroepen op november 20, 2015, van Code Project: http://www.codeproject.com/Articles/254091/Implementing-Microsoft-Unity-Design-Pattern

Oates, B. J. (2006). *Researching Information Systems and Computing.* London: SAGE Publications Ltd. Opgeroepen op september 14, 2015

RabbitMQ. (z.d.). *Documentation.* Opgeroepen op november 20, 2015, van RabbitMQ: https://www.rabbitmq.com/documentation.html

Sparx Systems. (z.d.). *Sparx Systems.* Opgeroepen op januari 4, 2016, van Enterprise Architect - UML Design Tools and UML CASE tools for software development: http://www.sparxsystems.com.au/products/ea/

Turnhout, K., Craenmehr, S., Holwerda, R., Menijn, M., Zwart, J.-P., & Bakker, R. (2013). *Triangulatie, een basis voor de onderzoeksleerlijn in ict en media onderwijs.* Opgeroepen op september 14, 2015, van NIOC - ontmoetingsplaats voor informatica-onderwijs: http://nioc.nl/archief/2013/wp-content/uploads/2013/12/nioc-paper-definitief-KvT.pdf

Vagrant. (z.d.). *Vagrant documentation.* Opgeroepen op november 20, 2015, van Vagrant: https://docs.vagrantup.com/v2/

Wikipedia. (z.d.). *Dependency injection.* Opgeroepen op november 23, 2015, van Wikipedia: https://nl.wikipedia.org/wiki/Dependency_injection

Woorden-boek.nl. (z.d.). *Voorspelling.* Opgeroepen op oktober 6, 2015, van Woorden-boek.nl: http://www.woorden-boek.nl/woord/voorspelling

9. Bijlagen

Plan van Aanpak

Predicting Server Flaws

Anthony Huebers

Plan van Aanpak

Predicting Server Flaws

Titel	Plan van Aanpak
Project/Onderwerp	Predicting Server Flaws
Versie	1.5
Status	Definitief
Datum	25-sep-2015
Bestand	Plan van Aanpak - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

9.1 Bijlage A - Plan van Aanpak

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	07-09-2015	Anthony Huebers	Creatie
1.1	Concept	10-09-2015	Anthony Huebers	Feedback Gert Jan Timmerman verwerkt.
1.2	Concept	11-09-2015	Anthony Huebers	Feedback Tom Nieuwenhuijs verwerkt.
1.3	Concept	14-09-2015	Anthony Huebers	Feedback Gert Jan Timmerman verwerkt.
1.4	Definitief	17-09-2015	Anthony Huebers	Feedback Tom Nieuwenhuijs verwerkt.
1.5	Definitief	25-09-2015	Anthony Huebers	Feedback Jelle Oosterkamp verwerkt.

Distributie

Versie	Status	Datum	Aan
1.0	Concept	10-09-2015	Gert Jan Timmerman (tijdelijke vervanger van Hylke Peek) en Tom Nieuwenhuijs
1.1	Concept	10-09-2015	Tom Nieuwenhuijs
1.2	Concept	11-09-2015	Gert Jan Timmerman (tijdelijke vervanger van Hylke Peek)
1.3	Concept	14-09-2015	Tom Nieuwenhuijs
1.4	Definitief	17-09-2015	Jelle Oosterkamp
1.5	Definitief	25-09-2015	Tom Nieuwenhuijs, Hylke Peek en Jelle Oosterkamp

Referenties

Code	Bron

Management samenvatting

Dit project is gestart omdat het regelmatig voorkomt dat er onverwachts een probleem ontstaat op een (machine op een) omgeving (van klanten) die beheerd wordt door Info Support. Als op voorhand gesignaleerd kan worden wanneer een (machine op een) omgeving problemen dreigt te krijgen, zou een werkelijk probleem mogelijk voorkomen kunnen worden.

Binnen het project wordt een (proof of concept) systeem gerealiseerd wat (patronen in) problemen kan herkennen en mogelijk problemen in de toekomst kan voorspellen. Het (proof of concept) systeem zal mogelijke problemen alleen signaleren en niet daadwerkelijk oplossen voor deze ontstaan.

Dit project heeft een doorlooptijd van 20 (effectieve) weken, verdeeld in drie fases, beginnende met een onderzoeksfase, gevolgd door een ontwikkelingsfase en tot slot een afrondingsfase.

- In de onderzoeksfase wordt een onderzoeksplan opgesteld en uitgevoerd wat resulteert in een onderzoeksrapport.
- In de ontwikkelingsfase wordt het (proof of concept) systeem gerealiseerd. Om tot een goed resultaat te komen wordt deze fase in iteraties uitgevoerd en wordt er gebruik gemaakt van de Agile Scrum methodiek. Naast het realiseren van het systeem worden er gedurende deze fase een functioneel en technisch ontwerp geschreven en bijgehouden.
- In de afrondingsfase vindt het schrijven van een scriptie, maken van een einddemo, de afstudeerpresentatie en de voorbereiding van de afstudeerzitting plaats.

Gedurende het project worden de volgende resultaten opgeleverd:

- Plan van Aanpak
- Voortgangsrapportages
- Onderzoeksplan
- Onderzoeksrapport
- Functioneel Ontwerp
- Technisch Ontwerp
- Programmacode
- Scriptie
- Afstudeerpresentatie
- Einddemo

Inhoudsopgave

1. Opdracht

1.1 Opdrachtgever en opdrachtnemer

De opdrachtnemer voor de in dit plan van aanpak beschreven project is:

Naam	: Anthony Huebers
Functie	: Afstudeerder
Organisatie	: Info Support
Telefoonnummer	: +31 (0)6 55 69 13 42
Emailadres	: anthony.huebers@infosupport.com

De opdrachtgever voor de in dit plan van aanpak beschreven project is:

Naam	: Tom Nieuwenhuijs
Functie	: Software Engineer
Organisatie	: Info Support
Telefoonnummer	: +31 (0)6 45 37 85 44
Emailadres	: tom.nieuwenhuijs@infosupport.com

1.2 Onderwijsinstelling

De onderwijsinstelling voor de in dit plan van aanpak beschreven project is (vertegenwoordigd door):

Naam	: Jelle Oosterkamp
Functie	: Docent
Organisatie	: Fontys Hogescholen
Telefoonnummer	: +31 (0)6 20 97 28 48
Emailadres	: j.oosterkamp@fontys.nl

1.3 Opdrachtdefinitie

1.3.1 Context

Een van de vele diensten die Info Support biedt is het op maat leveren van software aan klanten. Met leveren wordt niet alleen het implementeren van (eigen ontwikkelde of bestaande) software bedoeld, maar ook het beheer en draaiende houden van software (hosting).

Dit project wordt uitgevoerd voor de afdeling Professional Development Center (PDC) van Info Support. Deze afdeling faciliteert ontwikkelstraten en beheert (de virtuele machines en) de software die hiervoor nodig is.

In dit document wordt met “een (machine op een) omgeving” niet alleen de (virtuele) machine, maar een hele ontwikkelstraat met de daar onderliggende (virtuele) machines bedoeld.

1.3.2 Aanleiding

Info Support beheert omgevingen voor verschillende klanten. Deze omgevingen moeten idealiter non-stop beschikbaar zijn, dit is echter niet altijd het geval. Sommige problemen zorgen er zelfs voor dat de (machine op een) omgeving (tijdelijk) niet meer beschikbaar is. Er kunnen problemen optreden door een groot aantal oorzaken. Als op voorhand gemeld wordt wanneer een (machine op een) omgeving problemen dreigt te krijgen, zou een werkelijk probleem mogelijk voorkomen kunnen worden.

1.3.3 Probleemstelling

Het komt regelmatig voor dat er onverwachts een probleem ontstaat op een (machine op een) omgeving. Deze problemen vinden vaak op dezelfde of andere omgevingen opnieuw plaats. Info Support wil problemen kunnen voorspellen zodat ze niet meer onverwachts ontstaan.

1.3.4 Doelstelling

De opdrachtgever heeft als doel dat er proactief gehandeld wordt om te voorkomen dat er problemen ontstaan. Door afwijkend gedrag van een (machine op een) omgeving te melden, kunnen medewerkers van Info Support hier op inspelen om problemen mogelijk te voorkomen. Hierdoor moet het aantal problemen dat daadwerkelijk ontstaat verminderd worden.

De onderwijsinstelling en opdrachtnemer hebben als gezamenlijke doel het succesvol afronden van de afstudeerstage.

1.3.5 Resultaat

Om de doelstelling van de opdrachtgever te kunnen verwezenlijken, moet er een concept komen waarmee bewezen wordt dat problemen voorspeld kunnen worden. Het concept moet worden bewezen in de vorm van een (proof of concept) systeem dat problemen voorspelt. Om problemen te kunnen voorspellen moeten problemen of afwijkend gedrag herkend en gemeld worden. Door herkenning kan voorspeld worden of een probleem zich dreigt voor te doen. Wanneer een (eerder opgedaan) probleem dreigt te ontstaan, moet er een melding gegeven worden. Deze meldingen zijn zichtbaar in een dashboard. Op deze manier worden medewerkers van Info Support geïnformeerd en kunnen ze problemen mogelijk voorkomen.

Enkele voorbeelden van afwijkend gedrag waar voorspellingen opgemaakt kunnen worden:

- Uit systeeminformatie kan herleid worden wanneer een harde schijf vol dreigt te raken.
- Uit (de inhoud van) logbestanden kan herleid worden of een applicatie dreigt te crashen, omdat benodigde resources (zoals bijvoorbeeld een database) slecht of zelfs niet toegankelijk zijn.
- Uit beschikbaarheids- en reactie informatie kan herleid worden of een systeem toegankelijk is en hoe snel het systeem reageert op inkomende verzoeken.

Om de doelstelling van de onderwijsinstelling en opdrachtnemer te kunnen verwezenlijken, moet het resultaat voor de opdrachtgever behaald worden. Daarnaast moeten er nog documenten opgeleverd en verantwoord worden aan Fontys Hogescholen.

1.3.6 Effect

Wanneer er een bewezen concept beschikbaar is waarmee problemen voorspelt kunnen worden, kan de doelstelling van de opdrachtgever om proactief te handelen en mogelijk problemen te verminderen behaald worden. Het proof of concept kan gebruikt worden om het concept (op langer termijn) te testen. Wanneer het concept goed genoeg werkt, kan het in een volgend project uitgewerkt en geïmplementeerd worden in de organisatie om problemen te voorspellen.

Wanneer van het concept een systeem gerealiseerd is dat problemen voorspelt, zal er naast de doelstelling van de opdrachtgever ook een neveneffect zijn. Het (tijdelijk) neveneffect van dit resultaat is dat klanten hierdoor in de toekomst minder last zullen ondervinden van problemen omdat deze door eerdere meldingen sneller opgelost of voorkomen kunnen worden. Klanten zullen daardoor meer tevreden worden over de stabiliteit van de (machine op een) omgeving. Dit neveneffect verdwijnt na een periode omdat klanten dan gewent raken aan de nieuwe stabiliteit.

Wanneer naast het resultaat voor de opdrachtgever ook het resultaat voor de onderwijsinstelling behaald wordt, zal het doel om de afstudeerstage succesvol af te ronden bereikt zijn.

1.4 Afbakening

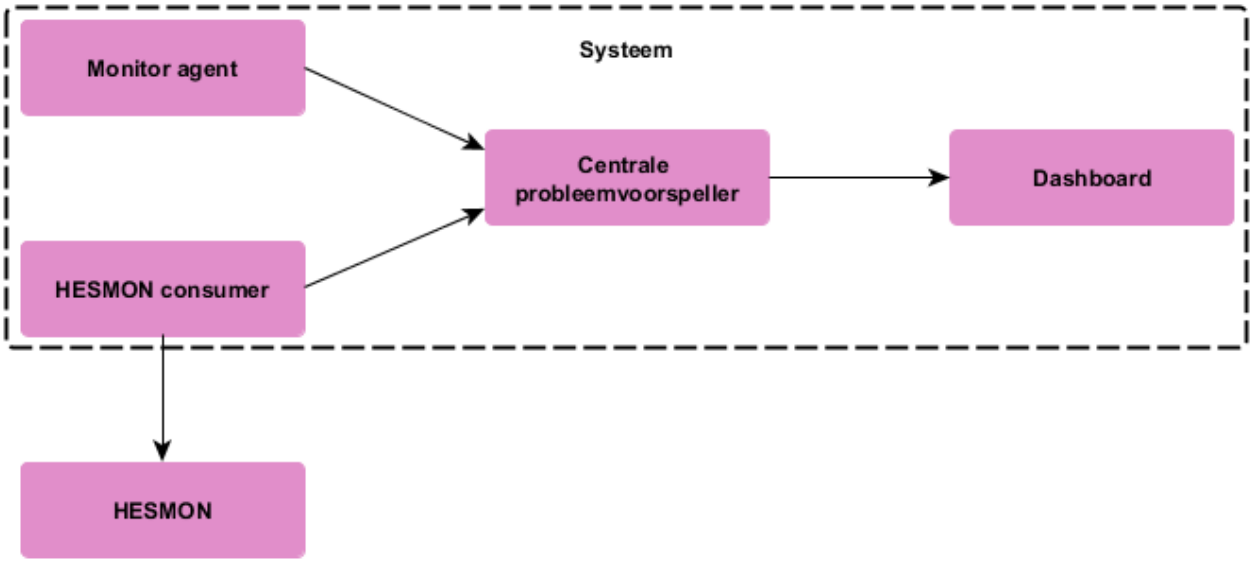
Het resultaat van dit project zal een beschreven concept zijn dat bewezen wordt door middel van een proof of concept. Het (proof of concept) systeem zal mogelijke problemen alleen melden en niet daadwerkelijk oplossen. Daarnaast hoef het niet direct inzetbaar te zijn binnen te organisatie.

Binnen het proof of concept wordt er gebruik gemaakt van bestaand component, een applicatie genaamd HESMON. Dit is een applicatie die de beschikbaarheid en reactietijd van meerdere (machines op meerdere) omgevingen meet en bewaart.

Het (proof of concept) systeem is verdeeld in vier componenten:

1. Een monitor agent die op meerdere (machines op meerdere) omgevingen geïmplementeerd wordt en die (de inhoud van) logbestanden en systeeminformatie verstuurt naar de centrale probleemvoorspeller.
2. Een HESMON consumer die van meerdere HESMON applicaties de beschikbaarheids- en reactie informatie van meerdere (machines op meerdere) omgevingen onttrekt en verstuurt naar de centrale probleemvoorspeller.
3. Een centrale probleemvoorspeller die van meerdere monitor agents en meerdere HESMON applicaties informatie ontvangt en bewaart. Deze (gecombineerde) informatie bestaande uit (inhoud van) logbestanden, systeem-, beschikbaarheids- en reactiesnelheidinformatie wordt vervolgens gebruikt om analyses en voorspellingen op toe te passen. Signaleringen die dit component maakt, worden kenbaar gemaakt aan het dashboard.
4. Het dashboard geeft interne medewerkers van Info Support inzicht in de signaleringen die gemaakt zijn door de centrale probleemvoorspeller.

In *Figuur 1 - Context-diagram* staan de besproken componenten en hun relaties met elkaar beschreven.



Figuur 1 - Context-diagram

Om tot een goed concept te komen is het belangrijk dat er onderzoek word gedaan en zullen de volgende resultaten opgeleverd worden ter ondersteuning:

1. Plan van Aanpak (dit document)
2. Onderzoeksplan
3. Onderzoeksrapport
4. Functioneel Ontwerp
5. Technisch Ontwerp
6. Scriptie
7. Voortgangsrapportages
8. Afstudeerpresentatie

1.5 Afhankelijkheden

Dit project kent geen afhankelijkheden omdat er geen externe projecten of leveranciers zijn betrokken.

1.6 Kwaliteitseisen

Dit project kent de volgende gestelde kwaliteitseisen:

1. De ontwikkelde programmacode voldoet aan de richtlijnen van Info Support. Deze worden door Info Support beschikbaar gesteld aan interne medewerkers.
2. De snelheid van het voorspellen van problemen vind realtime plaats met een zo kort mogelijke vertraging.
3. Alle documentatie is geschreven in correct Nederlands.
4. Het project moet van HBO niveau zijn om de afstudeerstage met succes af te ronden.

1.7 Uitgangspunten

Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande uitgangspunten van toepassing:

1. Het systeem wordt ontwikkeld in de programmeertaal C# met behulp van het .NET framework.
2. Communicatie van buitenaf naar het systeem moet via het HTTPS protocol gebeuren. Hiervoor is gekozen omdat HTTPS het enige veilige protocol is dat op alle (machines op) omgevingen beschikbaar is.
3. Analyses en voorspellingen worden gemaakt in de programmeertaal R (ontwikkeld voor statistische doeleinden).
4. De HESMON applicatie moet gebruikt worden voor het verkrijgen van beschikbaarheids- en reactie informatie van applicaties op (machines op) een omgeving.

1.8 Randvoorwaarden

Aan uitvoering van de in dit plan van aanpak beschreven delen zijn de volgende randvoorwaarden gesteld:

1. De opdrachtnemer werkt op locatie bij Info Support in Veenendaal (hoofdkantoor).

2. Risicomanagement

Voor uitvoering van de in dit plan van aanpak beschreven delen zijn de onderstaande risico's onderkend. Bij de risico's zijn de oorzaken en bijbehorende maatregelen ter beheersing opgenomen.

Nr	Risico omschrijving				
	Oorzaak	**Maatregel**	**P/S**	**Wie**	**Wanneer**
1	**Door beperkte onderzoekservaring ontstaat er tijdsgebrek**				
	Het resultaat van het onderzoek is niet bruikbaar of toepasbaar	Uitloopfase introduceren die één sprint in de ontwikkelingsfase vervangt	P	Opdracht-nemer	Bij niet bruikbaar of toepasbaar onderzoeksresultaat
2	**Door het ontbreken van kennis van de programmeertaal R is het mogelijk dat er eisen en wensen niet gerealiseerd kunnen worden**				
	De opdrachtnemer heeft nog nooit gewerkt met de programmeertaal R	De opdrachtnemer volgt een driedaagse training over de programmeertaal R	P	Opdracht-nemer	Voor het starten van de ontwikkelingsfase
3	**Niet alle gewenste problemen kunnen voorspeld worden**				
	De problemen kunnen niet worden voorspelt aan de hand van afwijkend gedrag	De opdrachtnemer bewijst dat de problemen niet voorspelbaar zijn	S	Opdracht-nemer	Bij het niet kunnen voorspellen van problemen
4	**De beschikbaarheid en toegankelijkheid van (informatie verstrekt door) de HESMON applicatie of (machines op) omgevingen**				
	Een (technische) storing	Alle verkregen beschikbaarheid en reactie informatie worden centraal bewaart. Eerder verkregen gegevens kunnen tijdelijk gebruikt worden	P	Opdracht-nemer	Wanneer de HESMON applicatie niet beschikbaar of toegankelijk is
	Een (technische) storing	Alle verkregen systeem informatie en (inhoud van) logbestanden worden centraal bewaart. Eerder verkregen gegevens kunnen tijdelijk gebruikt worden	P	Opdracht-nemer	Wanneer een (machine op) omgeving niet beschikbaar of toegankelijk is
5	**Opdrachtgever of onderwijsinstelling hebben tegenstrijdige wensen**				
	De opdrachtgever of onderwijsinstelling heeft extra wensen die de doelstelling van de ander tegenwerkt	De procesbegeleider om hulp vragen. Indien nodig om de tafel met opdrachtgever en stakeholder	S	Opdracht-nemer	Bij tegenstrijdige wensen

P/S = Preventief of Schade beperkend

3. Aanpak

3.1 Fasering

Dit project kent drie fases, beginnende met een onderzoeksfase, gevolgd door een ontwikkelingsfase en tot slot een afrondingsfase.

3.1.1 Onderzoeksfase

In de onderzoekfase wordt er vooronderzoek gedaan naar technieken die mogelijk gebruikt kunnen worden in het project. De vraag "*Hoe kunnen aan de hand van (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie problemen voorspeld worden?"* staat in dit onderzoek centraal.

Om de hoofdvraag te kunnen beantwoorden zijn de volgende deelvragen opgesteld:

1. Hoe verkrijg je (de inhoud van) logbestanden en systeeminformatie van een (machine op een) omgeving op een centrale plek?
2. Welk NoSQL oplossing is geschikt voor de centrale opslag van de gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie?
3. Hoe herken je afwijkend gedrag uit (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie?
4. Hoe kun je aan de hand van afwijkend gedrag problemen herkennen zodat deze voorspeld kunnen worden?

Deze fase is getimeboxed, zodat er tijd genoeg overblijft voor de ontwikkelingsfase. Aan het einde van deze fase worden de volgende resultaten opgeleverd:

- Onderzoeksplan
- Onderzoeksrapport

3.1.2 Ontwikkelingsfase

In de ontwikkelingsfase wordt het te ontwikkelen systeem gerealiseerd. Dit kan zijn door de eventuele implementatie van bestaande software, maar ook door het schrijven van (progammacode van) nieuwe software. Opgedane kennis uit de onderzoeksfase wordt in deze fase toegepast. Daarnaast kan door het onderzoek de opdrachtnemer beter inschatten hoe en wat voor (type) problemen voorspelt kunnen worden.

Gedurende de ontwikkelingsfase zal het (proof of concept) systeem dat problemen kan voorspellen in iteraties gerealiseerd worden. De exacte invulling en planning hiervan gebeurt volgens Agile Scrum methodiek en wordt nog nader bepaald.

Aan het einde van deze fase worden de volgende resultaten opgeleverd:

- Functioneel Ontwerp
- Technisch Ontwerp
- Proof of concept

3.1.3 Afrondingsfase

In de afrondingsfase wordt er tijd besteed aan de voorbereiding van het afsluiten van de afstudeerstage. In deze fase wordt er een scriptie geschreven en wordt een afstudeerpresentatie gemaakt en voorbereid. De afstudeerpresentatie zal een korte schermopname bevatten die de werking van het (proof of concept) systeem toont. Er is gekozen voor een einddemo in de vorm van een korte schermopname, omdat er beperkte tijd is tijdens de afstudeerzitting en het mogelijk is dat het (proof of concept) systeem alleen functioneert binnen de omgeving van Info Support.

Aan het einde van deze fase worden de volgende resultaten opgeleverd:

- Scriptie
- Afstudeerpresentatie
- Einddemo in de vorm van een korte schermopname

3.2 Ontwikkelmethodiek

Voor de onderzoeksfase en afrondingsfase zal er gebruik gemaakt worden van de timeboxing techniek. Hiervoor is gekozen omdat het belangrijk is dat er tijd gereserveerd wordt voor vooronderzoek en afronding van het project.

Aan het begin van de onderzoeksfase wordt een onderzoeksplan geschreven waarin beschreven staat hoe het onderzoek in de gekaderde tijd wordt uitgevoerd. Door een strak tijdkader wordt de diepgang van het onderzoek beperkt, maar is er wel genoeg ruimte om de beste bevindingen toe te passen.

Door een afrondingsfase te introduceren, kan tegen het einde van het project de focus gelegd worden op de eindpresentatie. Hiermee wordt de kans de doelstelling om de afstudeerstage te behalen vergroot.

Gedurende de ontwikkelingsfase wordt er een aangepaste versie van Agile Scrum gebruikt als ontwikkelmethodiek. Het element daily standup en de rol Scrum master zijn uit de methodiek gehaald, omdat het systeem door een eenmansteam ontwikkeld wordt.

Het product backlog element van Agile Scrum wordt gebruikt voor het bijhouden van een lijst van eisen en wensen (ook wel product backlog items genoemd) van het te ontwikkelen systeem. Gedurende het project kunnen hier wensen door de opdrachtgever aan toegevoegd worden. De opdrachtgever vervult de zogeheten product owner rol en bepaalt vervolgens ook de prioritering van deze lijst.

Het sprint element wordt gebruikt om in meerdere iteraties het systeem te realiseren. Door in sprints (met een duur van twee weken) te werken, blijft de opdrachtgever nauw betrokken bij de vooruitgang van het systeem. Bij het begin van iedere sprint wordt in overleg met de product owner een planning gemaakt welke product backlog items in de betreffende sprint gerealiseerd moeten worden. Halverwege iedere sprint worden het (bijgewerkte) functioneel en technische ontwerp opgeleverd. Aan het eind van iedere sprint worden de nieuwe ontwikkelingen aan het systeem gepresenteerd in de vorm van een demo.

Het sprint review/retrospective element wordt na iedere sprint gebruikt als evaluatiemoment. Hierin bespreken de opdrachtnemer en opdrachtgever wat inhoudelijk (review) en procesmatig (retrospective) goed en fout ging in de desbetreffende sprint. Eventuele verbeterpunten kunnen dan meegenomen worden in de volgende sprint.

Het definition of done element (zie 4.5.1) wordt gebruikt om in vast te leggen wanneer resultaten succesvol afgerond zijn.

9.1 Bijlage A - Plan van Aanpak

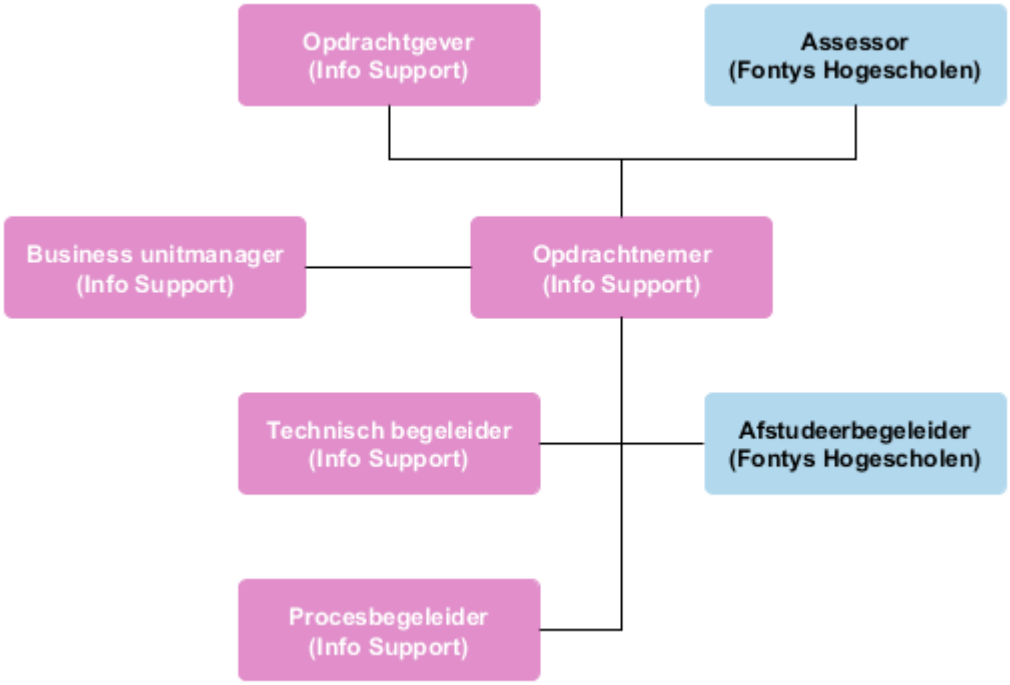
Agile Scrum is gekozen als ontwikkelmethodiek, omdat dit een methodiek is waarbij in meerdere iteraties het systeem ontwikkeld wordt. Door het systeem in meerdere iteraties te ontwikkelen, is er iedere twee weken een plan, demo en evaluatiemoment. Hierdoor is de kans dat het project slaagt groter, omdat de opdrachtgever nauw betrokken blijft en er continu ruimte is voor wijzingen en verbeteringen.

Iteratief werken kan ook door middel van andere methodieken. Er is echter gekozen voor de Agile Scrum methodiek omdat deze genoeg elementen bevat die toegepast kunnen worden in dit project. Daarnaast hebben alle betrokkenen van het project ervaring met Agile Scrum.

4. Beheersaspecten

4.1 Organisatie

4.1.1 Normstelling



Figuur 2 - Organisatiestructuur

De opdrachtgever is verantwoordelijk voor het verstrekken van de middelen aan de opdrachtnemer die nodig zijn om het project te laten slagen. Daarnaast is de opdrachtgever ook verantwoordelijk voor het prioriteren van de backlog in de ontwikkelingsfase en het accorderen van keuzes die gemaakt worden in de loop van het project. Tom Nieuwenhuijs, Software Engineer bij Info Support is hiervoor aangesteld.

De assessor is verantwoordelijk voor het beoordelen van de documenten die opgeleverd worden door de opdrachtnemer. René van der Heijden, Docent bij Fontys Hogescholen is hiervoor aangesteld.

De business unitmanager volgt de voortgang van het afstuderen. De unitmanager is daarom op de achtergrond betrokken bij het project om de voortgang van de opdrachtnemer te volgen. Mark Klabbers, Business Unit Manger bij Info Support vervult deze rol.

De opdrachtnemer is verantwoordelijk voor het uitvoeren van het project en het realiseren en opleveren van de op te leveren resultaten. Om tot een goed resultaat te komen, wordt er ook vooronderzoek gedaan. Anthony Huebers, Afstudeerder bij Info Support is hiervoor aangesteld.

De technisch begeleider is verantwoordelijk voor het beschikbaar stellen van technische kennis en ondersteuning aan de opdrachtnemer. Hylke Peek, Software Engineer bij Info Support is hiervoor aangesteld.

De afstudeerbegeleider is verantwoordelijk voor het geven van feedback op de documenten die opgeleverd moeten worden aan de assessor. Daarnaast is deze verantwoordelijk voor het begeleiden van de opdrachtnemer bij problemen die het behalen van de afstudeerstage belemmeren. Jelle Oosterkamp, Docent bij Fontys Hogescholen is hiervoor aangesteld.

De procesbegeleider is verantwoordelijk voor het ondersteunen van de opdrachtnemer bij niet-technische problemen. Daarnaast zal deze de voortgang en persoonlijke ontwikkeling van de opdrachtnemer begeleiden. Marieke Keurntjes, Opleidingsadviseur bij Info Support is hiervoor aangesteld.

4.2 Informatie

4.2.1 Normstelling

Gedurende het project worden alle bevindingen, gemaakte keuzes en voortgang door de opdrachtnemer gedocumenteerd. Over de inhoud van de documenten wordt de opdrachtgever mondeling of per mail op de hoogte gehouden door de opdrachtnemer.

Aan het einde van het project levert de opdrachtnemer een concept op hoe problemen voorspelt kunnen worden.

Voor het starten van de onderzoeksfase wordt een lijst opgeleverd van applicaties en onderdelen van een (machine op een) omgeving die gemonitord moeten worden. Daarnaast verleent de opdrachtgever in het begin van de ontwikkelingsfase toegang tot een of meerdere (machines op een of meerdere) (test)omgevingen die gebruikt kunnen worden voor monitoring.

Wanneer er door de opdrachtnemer behoefte is aan extra informatie van de opdrachtgever of het beschikbaar stellen van technische kennis en ondersteuning van de technisch begeleider, gebeurt dit per mail of mondeling. Hierbij is een maximale responstijd van twee volledige werkdagen.

4.2.2 Voortgangscontrole

In het periodiek overleg tussen de opdrachtnemer en opdrachtgever zullen de bevindingen, gemaakte keuzes en voortgang worden besproken.

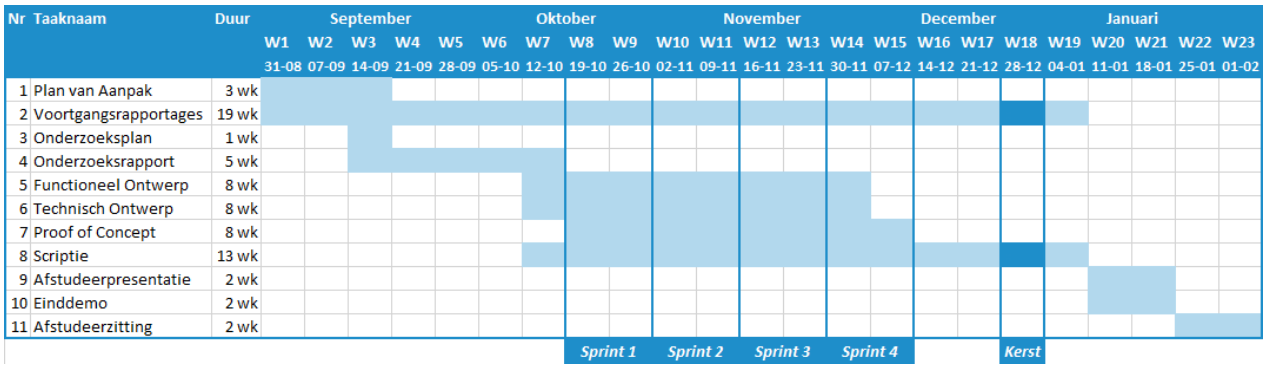
Het concept wordt door de opdrachtnemer aan de opdrachtgever gepresenteerd. Het concept wordt vertegenwoordigd en ondersteund door een (proof of concept) systeem, Onderzoeksverslag, Functioneel- en Technisch Ontwerp samen.

De opdrachtgever verzorgt de gevraagde informatie en verleent indien nodig toegang tot extra (machines op een of meerdere) (test)omgevingen wanneer de opdrachtnemer hier na vraagt.

4.3 Tijd

4.3.1 Normstelling

Dit project heeft een doorlooptijd van 20 (effectieve) weken, beginnende op 31-08-2015 tot en met 05-02-2016. De laatste weken van dit project zijn gereserveerd voor de afstudeerzitting. Gedurende de weken die gereserveerd zijn voor de afstudeerzitting wordt er niet meer gewerkt aan het project. De definitieve datum van de afstudeerzitting moet nog door de assessor bekend gemaakt worden aan de opdrachtnemer.



Figuur 3 – Gantt chart

4.3.2 Voortgangscontrole

Gedurende het project wordt de voortgang bewaakt door periodieke gesprekken met de opdrachtgever, technisch begeleider en procesbegeleider. Daarnaast wordt er eenmaal per twee weken een voortgangsrapportage verstuurt naar de opdrachtgever en alle begeleiders.

4.4 Middelen

4.4.1 Normstelling

De onderstaande middelen worden door de opdrachtgever beschikbaar gesteld aan de opdrachtnemer:

1. Er is een ontwikkelomgeving beschikbaar met daarop een recente versie (2012 of nieuwer) van Visual Studio.
2. Er moet een Team Foundation Server beschikbaar zijn.

4.4.2 Voortgangscontrole

Wanneer middelen ontbreken of onvoldoende blijken te zijn, meldt de opdrachtnemer dit direct aan de opdrachtgever. De opdrachtgever zorgt ervoor dat de middelen waarover van tevoren overeenstemming bereikt is, beschikbaar worden gesteld.

4.5 Kwaliteit

4.5.1 Normstelling

De definition of done voor documenten is als volgt:

- Het document is voor feedback naar de opdrachtgever, technisch- en afstudeerbegeleider verstuurt
- Ontvangen feedback is verwerkt in het document
- Het document bevat geen spelling – en grammatica fouten
- Het definitieve document is als PDF opgeslagen en verstuurt naar alle betreffende betrokkenen.

De definition of done voor backlog items is als volgt:

- De functionaliteit die beschreven staat in het backlog item is gerealiseerd.
- De acceptatiecriteria die beschreven staat in het backlog item zijn nagekomen.
- De programmacode is voorzien van unittesten, die minimaal 70% van de programmacode dekken en slagen.
- De geschreven programmacode en testen voldoet aan de richtlijnen van Info Support.
- De programmacode en unittesten zijn opgeslagen in Team Foundation Service.

4.5.2 Voortgangscontrole

Bij het opleveren van resultaten wordt de definition of done gecontroleerd. Dit gebeurt voor documenten door de begeleiders en voor backlog items door de opdrachtgever en technisch begeleider samen.

De afstudeerbegeleider controleert voortdurend of het project nog van correct niveau is. Wanneer dit niet meer van het gewenste niveau is, wordt dit per mail aan de opdrachtnemer, opdrachtgever en technisch begeleider gemeld.

De maximale vertraging die beschreven staat bij de kwaliteitseisen mag maximaal 1 minuut bedragen. Dit kan gemeten worden door de tijd waarop een gebeurtenis heeft plaats gevonden op een (machine op een) omgeving af te trekken van de tijd waarop de melding is gemaakt door het systeem.

4.6 Overlegvormen

In de onderzoeksfase van het project vindt er eenmaal per week overleg plaats in de vorm van een vergadering met de opdrachtgever. Gedurende de rest van het project is dit eenmaal per twee weken. Tijdens dit overleg wordt de voortgang van het project besproken en in de ontwikkelingsfase worden er sprints ingepland en geëvalueerd.

Eenmaal per twee weken vindt er met de technisch- en procesbegeleider apart een overleg plaats in de vorm van een vergadering. Tijdens dit overleg wordt de vooruitgang van het project en eventuele belemmeringen besproken.

De afstudeerbegeleider zal Info Support tweemaal bezoeken voor een overleg in de vorm van een evaluatiegesprek.

Eenmaal per zes weken vindt er een voortgangsgesprek plaats met de business unitmanager. In dit gesprek wordt de voortgang van het project kort en bondig besproken.

De stakeholder blijft op de hoogte via de (voortgangsrapportage verstuurt aan de) afstudeerbegeleider.

4.7 Communicatie

In onderstaand overzicht is de communicatie en verantwoordelijkheden tussen alle betrokkenen beschreven.

Fase/proces	Ondersteunen	Uitvoeren	Beslissen (goedkeuren)	Gebruiken (informeren)
Periodiek overleg	n.v.t.	ON	n.v.t.	OG, TB & PB
Bezoek afstudeerbegeleider	OG, TB & PB	ON	n.v.t.	AB
Voortgangsgesprek	n.v.t.	ON	n.v.t.	BM
Voortgangsrapportage	n.v.t.	ON	n.v.t.	OG, TB, PB & AB
Sprint planning	n.v.t.	ON	OG	OG
Sprint review / retrospective	n.v.t.	ON	OG	OG
Plan van Aanpak	n.v.t.	ON	OG, TB & AB	OG, AS, TB, PB & AB
Onderzoeksplan	n.v.t.	ON	OG & TB	OG, TB & PB
Onderzoeksrapport	n.v.t.	ON	OG & TB	OG, TB, PB & AB
Functioneel Ontwerp	n.v.t.	ON	OG & TB	OG & TB
Technisch Ontwerp	n.v.t.	ON	TB	OG & TB
Proof of Concept	n.v.t.	ON	OG & TB	OG & TB
Scriptie	n.v.t.	ON	OG, TB & AB	OG, AS, BM, TB, PB & AB

OG = Opdrachtgever
ON = Opdrachtnemer
AB = Afstudeerbegeleider
AS = Assessor
TB = Technisch begeleider
BM = Business Unitmanager
PB = Procesbegeleider

5. Oplevering

5.1 Normstelling

De gemaakte documenten maken gebruik van het Info Support 2014 template en worden in docx en (indien goedgekeurd in) PDF formaat bewaart op de (netwerkdrive genaamd) afstudeershare. Het template en de netwerkdrive worden beschikbaar gesteld door Info Support.

Standaard is ieder document een concept document. Om een concept document definitief te maken, wordt deze eerst door de opdrachtnemer verstuurt naar de opdrachtgever en technisch begeleider voor feedback. Nadat de opdrachtnemer de feedback heeft verwerkt, wordt het document ter goedkeuring verstuurt naar de opdrachtgever. Wanneer er meer opmerking zijn dan het aantal pagina of wanneer het document niet goed gekeurd is, wordt het document opnieuw verstuurt naar de opdrachtgever en technisch begeleider voor feedback.

Wanneer een document goedgekeurd is door de opdrachtgever, wordt deze voor feedback doorgestuurd naar de afstudeerbegeleider. Eventuele feedback wordt vervolgens verwerkt en teruggekoppeld per mail aan de opdrachtgever. Als de opdrachtgever akkoord gaat met de verwerkte feedback wordt het document gemarkeerd als definitieve versie. Wanneer de opdrachtgever niet akkoord gaat met verwerkte feedback wordt er een gesprek geplant tussen de opdrachtgever en afstudeerbegeleider om tot een gezamenlijk resultaat te komen.

De definitieve versie wordt vervolgens in PDF formaat bewaart en doorgestuurd naar de opdrachtgever, assessor en alle begeleiders. Het versturen van documenten gebeurt altijd per e-mail.

De product backlog en sprint planning worden door de opdrachtnemer in overleg met de opdrachtgever geregistreerd en bijgehouden in Team Foundation Server.

5.2 Voortgangscontrole

In onderstaand overzicht staan de op te leveren documenten, de uiterste oplevermomenten van elk document en de inleverdata beschreven.

Document	Oplevering	Aan
Plan van Aanpak	18-09-2015 (W3)	OG, TB, PB, AB & AS
Onderzoeksplan	25-09-2015 (W4)	OG, TB & PB
Onderzoeksrapport	23-10-2015 (W8)	OG, TB, PB & AB
Functioneel Ontwerp	11-12-2015 (W15)	OG & TB
Technisch Ontwerp	11-12-2015 (W15)	OG & TB
Proof of Concept	11-12-2015 (W15)	OG
Scriptie	08-01-2016 (W19)	OG, TB, PB, AB & AS

OG = Opdrachtgever
PB = Procesbegeleider
AS = Assessor
AB = Afstudeerbegeleider
TB = Technisch begeleider

5.3 Resultaten

De uitvoering van dit project resulteert in de onderstaande resultaten:

1. **Plan van Aanpak**
   Hierin staat beschreven hoe het project wordt uitgevoerd (dit document).
2. **Voortgangsrapportages**
   Dit is een (continu groeiende) rapportage van de voortgang van het project dat iedere twee weken rondgestuurd wordt aan alle belanghebbende van het project.
3. **Onderzoeksplan**
   Hierin staat beschreven wat er onderzocht wordt en hoe het vooronderzoek uitgevoerd wordt.
4. **Onderzoeksrapport**
   Dit is een rapportage van de bevindingen van het vooronderzoek met een conclusie welke technieken er gebruikt moeten gaan worden.
5. **Functioneel Ontwerp**
   Dit is een ontwerp waarin de functies die het (proof of concept) systeem moet gaan aanbieden beschreven staan en verduidelijk worden in klasse- en sequence diagrammen.
6. **Technisch Ontwerp**
   Dit is een ontwerp waarin alle technisch keuzes die gemaakt worden voor de realisatie van het (proof of concept) systeem beschreven. Daarnaast bevat het klasse-, sequence- en componenten diagrammen die als blauwdruk voor de ontwikkeling van het (proof of concept) systeem gebruikt kunnen worden.
7. **Proof of Concept**
   De door het opdrachtnemer gerealiseerd (proof of concept) systeem. De geïmplementeerde componenten en bijbehorende programmacode.
8. **Scriptie**
   Dit is een verslag waarin de uitvoering en resultaat van het project beschreven staat. Dit is een allesomvattend verslag waarin ook de bevindingen van het vooronderzoek beschreven staan.
9. **Afstudeerpresentatie**
   Een presentatie over het gehele proces en resultaat van het project waarmee de afstudeerstage afgerond kan worden.
10. **Einddemo**
    Korte schermopname van de werking van het (proof of concept) systeem.

Onderzoeksrapport

Predicting Server Flaws

Anthony Huebers

Onderzoeksrapport

Predicting Server Flaws

Titel	Onderzoeksrapport
Project/Onderwerp	Predicting Server Flaws
Versie	1.1
Status	Definitief
Datum	13-okt-2015
Bestand	Onderzoeksrapport - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

9.2 Bijlage B - Onderzoeksrapport

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	18-09-2015	Anthony Huebers	Creatie
1.1	Definitief	13-10-2015	Anthony Huebers	Feedback Tom Nieuwenhuijs en Jelle Oosterkamp verwerkt

Distributie

Versie	Status	Datum	Aan
1.0	Concept	7-10-2015	Tom Nieuwenhuijs, Hylke Peek en Jelle Oosterkamp
1.1	Definitief	13-10-2015	Tom Nieuwenhuijs en Hylke Peek

Referenties

Code	Bron

9.2 Bijlage B - Onderzoeksrapport

Management Samenvatting

Er is onderzocht hoe beschikbare gegevens gebruikt kunnen worden om voorspellingen te maken. De vraag "*Hoe kunnen aan de hand van (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie problemen voorspeld worden?*" stond in dit onderzoek centraal.

Dit onderzoek is voornamelijk door middel van literatuuronderzoek uitgevoerd.

Op basis van de onderzoeksresultaten blijkt dat het maken van voorspellingen van problemen die gekoppeld zijn aan gegevens uit systeem-, beschikbaarheids- en reactie informatie mogelijk is. Door middel van Time Series Analysis kan het ontstaan van afwijkend gedrag voor gegevens uit systeem-, beschikbaarheids- en reactie informatie voorspeld worden. Dit is een algoritme om tijd gedreven gegevens mee te analyseren. Door de verwachting te vergelijken met het daadwerkelijke gedrag wordt afwijkend gedrag bekend voordat problemen ontstaan.

Uit (de inhoud van) logbestanden is het ook mogelijk afwijkend gedrag te herkennen maar is dit vaak te laat om nog voorspellingen te kunnen maken. Dit betekend echter niet dat (de inhoud van) logbestanden onbruikbaar is. Door logberichten te groeperen per Severity kan de verwachte frequentie waarop bepaalde logberichten ontstaan vergeleken worden met de daadwerkelijke frequentie.

Tijdens het onderzoek is er ook aandacht besteed aan het zoeken en vergelijken van beschikbaar softwarepakketten voor het verkrijgen en centraliseren van de benodigde gegevens. Ook zijn de centrale opslagmogelijkheden binnen NoSQL onderzocht.

Omdat aan de hand van systeem-, beschikbaarheids- en reactie informatie het beste voorspellingen gemaakt kunnen worden, is het advies hierop de focus te leggen bij de realisatie van een concept.

9.2 Bijlage B - Onderzoeksrapport

Inhoudsopgave

9.2 Bijlage B - Onderzoeksrapport

1. Context

1.1 Over Info Support

Info Support is een specialist in het ontwikkelen, beheren en hosten van software op maat en het leveren van BI- en integratieoplossingen. Het bedrijf bestaat al sinds 1986 en is inmiddels van eenmansbedrijf uitgegroeid tot ruim 400 medewerkers met vestigingen verspreid over Nederland en België *(Info Support, z.d.)*.

Binnen Info Support is er een afdeling genaamd Professional Development Center (PDC). De afdeling PDC bestaat uit 11 medewerkers en faciliteert en beheert de ontwikkelstraat die door alle ontwikkelaars van Info Support gebruikt wordt. Het onderzoek wordt voor de afdeling PDC uitgevoerd.

1.2 Aanleiding

De omgevingen van klanten moeten idealiter non-stop beschikbaar zijn, dit is echter niet altijd het geval. Sommige problemen zorgen er zelfs voor dat de (machine op een) omgeving (tijdelijk) niet meer beschikbaar is. Er kunnen problemen optreden door een groot aantal oorzaken. Als op voorhand gemeld wordt wanneer een (machine op een) omgeving problemen dreigt te krijgen, zou een werkelijk probleem mogelijk voorkomen kunnen worden.

Om een hogere beschikbaarheid van omgevingen te creëren is daarom belangstelling in het voorspellen van problemen. Door problemen te voorspellen kan er proactief gehandeld worden om te voorkomen dat een probleem ontstaat.

2. Onderzoeksvragen

2.1 Afbakening

Het onderzoek kent de onderstaande afbakening:

- In dit onderzoek worden alleen (machines op) omgevingen gebruikt met het Windows of Linux besturingssysteem.
- Met systeeminformatie wordt het processor, geheugen, schijf en netwerk gebruik bedoelt.
- Voor de opslag van gegevens moet een NoSQL oplossingen gebruikt worden.
- De gekozen NoSQL oplossing moet uitvoerbaar zijn in Azure.

2.2 Onderzoeksvragen

Gedurende dit onderzoek moet het bekend worden hoe gegevens verkregen en centraal bewaard kunnen worden. Daarnaast moet het duidelijk worden welke informatie er uit gegevens gehaald kan worden en op welke manier deze informatie verkregen kan worden. Verder moet het duidelijk worden hoe de toekomst voorspeld kan worden aan de hand van informatie uit het verleden.

Om dit allemaal te kunnen beantwoorden staat de vraag "*Hoe kunnen aan de hand van (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie problemen voorspeld worden?*" in dit onderzoek centraal.

Om de hoofdvraag te kunnen beantwoorden zijn de volgende deelvragen opgesteld:

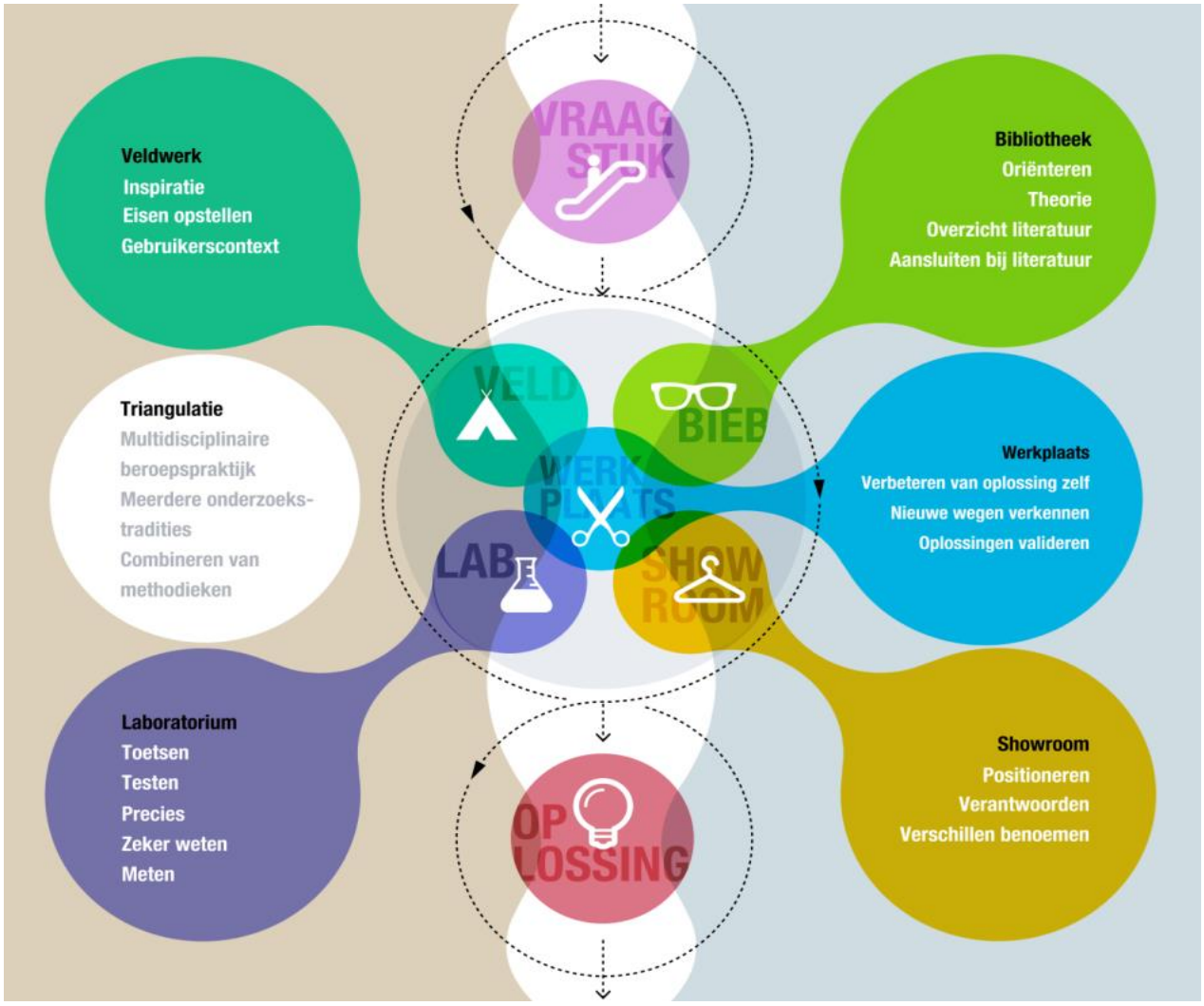
1. Hoe verkrijg je (de inhoud van) logbestanden en systeeminformatie van een (machine op een) omgeving op een centrale plek?
2. Welk NoSQL oplossing is geschikt voor de centrale opslag van de gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie?
3. Hoe herken je afwijkend gedrag uit (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie?
4. Hoe kun je aan de hand van afwijkend gedrag problemen herkennen zodat deze voorspeld kunnen worden?

De eerste en tweede deelvraag zijn opgesteld omdat er een concept gerealiseerd moet worden. Om een proof of concept te kunnen maken is de technische kennis die deze deelvragen verschaffen essentieel.

3. Aanpak

3.1 Methodiek

Bij de uitvoering van het onderzoek wordt het triangulatie-framework gebruikt als hulpmiddel om te bepalen hoe het onderzoek uitgevoerd moet worden. Door verschillende methoden die het triangulatie-framework biedt te combineren kunnen onderzoeksvragen beter beantwoord worden *(Oates, 2006).* De verschillende methoden zijn verdeeld in vijf onderzoeksruimtes die te zien zijn in ***Error! Reference source not found.***.



Figuur 1 - Triangulatie framework (Middelkoop, 2015)

3.1.1 Onderzoeksruimtes

Iedere onderzoeksruimte heeft een eigen aanpak en een bijbehorende verzameling methodieken en technieken. Hieronder staat in het kort beschreven welke focus iedere onderzoeksruimte heeft *(Turnhout, et al., 2013)*.

- Bieb richt zich op het verkrijgen van beschikbare literatuur en het voortbouwen op eerder uitgevoerd onderzoek.
- Werkplaats richt zich op het experimenteren met oplossingen om zo tot nieuwe bevindingen te komen.
- Showroom richt zich op de herbruikbaarheid van oplossingen en het vergelijken van verschillende oplossingen.
- Laboratorium richt zich op het toetsen en meten van oplossingen.
- Veld richt zich op het verdiepen in het toepassingsdomein door middel van contact met betrokkenen personen.

3.2 Uitvoering

Dit onderzoek heeft een strak tijdkader van vijf weken. Om alle deelvragen te kunnen beantwoorden binnen het strakke tijdkader is er gekozen om een vaste periode te definiëren voor iedere deelvraag en ruimte te reserveren voor conclusies te trekken uit de gevonden bevindingen. Aan het einde is dan uitloop mogelijk wanneer een deelvraag niet genoeg diepgang heeft.

1. *Hoe verkrijg je (de inhoud van) logbestanden en systeeminformatie van een (machine op een) omgeving op een centrale plek?*

Deze vraag wordt beantwoord door methodieken uit de bibliotheek- en veldwerk onderzoeksruimte. Door middel van gesprekken met de opdrachtgever wordt duidelijk welke logbestanden (van welke applicaties) er gebruikt moeten gaan worden. Vervolgens wordt er door middel van literatuuronderzoek onderzocht hoe de gevraagde informatie verkregen en gecentraliseerd kan worden met behulp van bestaande tools.

2. *Welk NoSQL oplossing is geschikt voor de centrale opslag van de gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie?*

Deze vraag wordt beantwoord door methodieken uit de bibliotheek- en showroom onderzoeksruimte. Door middel van literatuuronderzoek worden onderzocht welke technieken er bestaan en welke voor en nadelen iedere techniek bevat. Daarnaast word er onderzocht wat er mogelijk is in Azure omtrent NoSQL oplossingen.

3. *Hoe herken je afwijkend gedrag uit (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie?*

Deze vraag wordt beantwoord door methodieken uit de bibliotheek- en veldwerk onderzoeksruimte. Door middel van literatuuronderzoek in combinatie met een interview met een expert (binnen Info Support) over data analyse wordt er onderzocht hoe logbestanden, systeem-, beschikbaarheids- en reactie informatie geanalyseerd kunnen worden om afwijkend gedrag te herkennen.

4. *Hoe kun je aan de hand van afwijkend gedrag problemen herkennen zodat deze voorspeld kunnen worden?*

Deze vraag wordt beantwoord door methodieken uit de bibliotheek- en veldwerk onderzoeksruimte. Door middel van literatuuronderzoek in combinatie met een interview met een expert (binnen Info Support) over data analyse wordt er onderzocht hoe door middel van afwijkend gedrag problemen voorspeld herkent kunnen worden. Om dit te bereiken wordt er onderzocht wat problemen zijn en hoe deze gekoppeld kunnen worden aan afwijkend gedrag. Vervolgens wordt er onderzocht hoe het herkennen van problemen kan leiden tot voorspellen.

4. Resultaten

4.1 Gegevens verzamelen

Om voorspellingen te kunnen maken, moeten gegevens uit het verleden en het heden beschikbaar zijn. Bij het beantwoorden van deze deelvraag is er door middel van literatuuronderzoek onderzocht met welke beschikbare softwarepakketten gegevens verkregen en gecentraliseerd kunnen worden. Door beschikbare softwarepakketten te vergelijken kan een advies gegeven worden.

In dit hoofdstuk staat de deelvraag "*Hoe verkrijg je (de inhoud van) logbestanden en systeeminformatie van een (machine op een) omgeving op een centrale plek?*" centraal.

4.1.1 Logbestanden

Logbestanden zijn tekstbestanden waarin staat beschreven welke evenementen er hebben plaatsgevonden op een systeem *(Christensson, 2010)*. De manier waarop evenementen worden beschreven verschilt echter per logbestand. Ondanks dat er geen echte richtlijnen zijn hoe een logbericht opgebouwd moet worden, zijn er wel een aantal wereldwijd geaccepteerde standaarden, zoals bijvoorbeeld Syslog en (NCSA) Common Log Format.

4.1.1.1 Syslog

Syslog is een standaard die gehanteerd wordt door (veel applicaties uitgevoerd op) het Linux en Unix platform *(Logentries, z.d.)*. Binnen de Syslog standaard kan normaliter de structuur van logberichten opgedeeld worden in vier onderdelen:

- Date, datum en tijd wanneer een evenement heeft plaatsgevonden.
- Facility, beschrijft tijdens welk soort proces (van een applicatie) een evenement heeft plaatsgevonden.
- Severity (level), beschrijft de ernst van het evenement.
- Message, beschrijft in de vorm van tekst of tags een evenement.

In de Syslog standaard staat beschreven welke numerieke waarde een Facility en Severity representeert. Deze Severity waardes zijn onder te verdelen in acht levels (zie *Tabel 1 - Beschikbare Severity in Syslog standaard*). Door middel van een formule kan aan de hand van de waardes van de Facility en Severity een prioriteit voor ieder logbericht berekend worden. De prioriteit wordt berekend door de waarde van de Facility te vermenigvuldigen met acht (het aantal Severity waardes) en op te tellen bij de waarde van de Severity *(Gerhards, 2009)*.

Tabel 1 - Beschikbare Severity in Syslog standaard

Severity (level)	Omschrijving
Emergency / Fatal	Een fout dat er voor zorgt dat het systeem onbruikbaar wordt.
Alert	Een fout waarop onmiddellijk actie moet worden ondernomen.
Critical	Een kritische foutmelding
Error	Een (normale) foutmelding
Warning	Een waarschuwing, bevat vaak informatie over verwachte fouten.
Notice	Een normale maar vermeldbare situatie heeft plaats gevonden.
Information	Een informatie bericht
Debug	Een debug bericht

4.1.1.2 (NCSA) Common Log Format

(NCSA) Common Log Format is een standaard die gehanteerd wordt door webservers. Binnen deze standaard bevat een logbericht altijd de volgende onderdelen (in benoemde volgorde) *(Loganalyzer, z.d.)*:

1. (Remote) host, het IP adres van de cliënt die een verzoek indient bij de webserver.
2. User-identifier, de identiteit van de cliënt.
3. Username, de identiteit van de gebruiker geverifieerd door de cliënt. Dit wordt door de webserver gebruikt voor authenticatie.
4. Date, datum en tijd wanneer het verzoek is ingediend.
5. Request, het HTTP verzoek gemaakt door de cliënt beschreven in drie delen:
    1. De opgevraagde bron (bijvoorbeeld index.html)
    2. De gebruikte HTTP methode (bijvoorbeeld POST of GET)
    3. De gebruikte versie van het HTTP protocol (bijvoorbeeld HTTP/1.1)
6. Statuscode, de HTTP status code die de cliënt heeft ontvangen op het verzoek.
7. Bytes, de lengte van het bericht in bytes dat de cliënt heeft ontvangen op het verzoek.

4.1.1.3 Overige formaten

IIS, een veel gebruikte web service, hanteert het IIS Log File Format. Dit format bevat naast de onderdelen beschreven in Common Log Format *(Microsoft Technet, z.d.)* extra informatie zoals bijvoorbeeld de identiteit van de server en de lengte van het bericht in bytes dat de cliënt heeft verstuurd. Het is in ISS mogelijk de NCSA of W3C standaard te gebruiken in plaats van de IIS Log File Format.

De Windows Event Logs bevatten dezelfde vier onderdelen die beschreven zijn in de Syslog standaard. Daarnaast worden nog een hoop extra details in XML formaat bewaard. De XML gegevens zijn echter geen tekst, maar binaire (.evt of .evtx) bestanden *(Schuster, 2007)*.

Andere standaarden van logberichten worden niet genoemd in dit onderzoek, omdat deze wat betreft (basis) structuur veel lijken of gebaseerd zijn op een van de twee besproken standaarden.

4.1.2 Te monitoren softwarepakketten

4.1.2.1 Overzicht

In dit onderzoek is er gekozen om de logbestanden te analyseren van softwarepakketten die gebruikt worden in een (software)ontwikkelstraat van Java en .NET projecten. In *Tabel 2 - Standaard log locatie* zijn de standaard log locaties van ieder softwarepakket voor zowel het Windows als Linux platform beschreven.

Tabel 2 - Standaard log locatie

Softwarepakket	Standaard log locatie	
	Windows	Linux
Team Foundation Server (TFS)	C:\ProgramData\Microsoft\Team Foundation\Server Configuration\Logs	N.v.t.
Jenkins	In de root van de Jenkins installatie en in de Tomcat installatiemap.	/var/log/jenkins/jenkins.log /var/log/tomcat/catalina.out
JIRA	c:\WINDOWS\system32\atlassian-jira.log en de root van de JIRA home map.	In het bestand logs/catalina.out van de JIRA installatie map en in de JIRA home map.
Stash	In de map logs van de Stash installatie map en in de Stash home map.	In de map logs van de Stash installatiemap en in de Stash home map.
SonarQube	logs/sonar.log van SonarQube home directory	logs/sonar.log van SonarQube home directory
Nexus	In het bestand logs/nexus.log in de Nexus Work Directory map	In het bestand logs/nexus.log in de Nexus Work Directory map
Confluence	c:/confluence/data/logs/atlassian-confluence.log en de root van de Confluence home map.	In het bestand logs/catalina.out van de Confluence installatie map en in de Confluence home map.
MSSQL	%Program-Files%\Microsoft SQL Server\MSSQL.<MSSQLVERSIE>\MSSQL\LOG\ERRORLOG of Windows Event Log	N.v.t.
Internet Information Services (IIS)	%SystemDrive%\inetpub\logs\LogFiles of Windows Event Log	N.v.t.
NGINX	C:\nginx\logs\error.log	/var/log/nginx/error.log
Sharepoint	C:\program files\common files\microsoft shared\web server extensions\<sharepointversie>\LOGS\	N.v.t.
Docker	In de root van de Docker installatie en in de root van iedere container. *	/var/log/docker.log en in de root van iedere container. *

** Van iedere container wordt alleen de console output (stdout) van het betreffende logbestand weggeschreven. Logbestanden gegeneerd door applicaties in de container staan opgeslagen in de container.*

4.1.2.2 Relaties

Tabel 3 - Relaties tussen softwarepakket(ten)

Softwarepakket	Gebruikt of heeft een relatie met softwarepakket(ten)
Team Foundation Server (TFS)	MSSQL
Jenkins	Stash, SonarQube en Nexus *
JIRA **	MSSQL
Stash **	MSSQL
SonarQube	MSSQL
Nexus	N.v.t.
Confluence **	MSSQL
MSSQL	N.v.t.
Internet Information Services (IIS)	N.v.t.
NGINX	N.v.t.
Sharepoint	IIS en MSSQL
Docker	N.v.t.

** Ondanks dat er een relatie is tussen de genoemde softwarepakketten, blijft Jenkins onafhankelijk van de gerelateerde softwarepakketten functioneren.*

*** Deze softwarepakketten kunnen aan elkaar gekoppeld worden door middel van een web portal, maar blijven onafhankelijk van elkaar functioneren.*

4.1.3 Systeeminformatie

Systeeminformatie is een allesomvattend woord voor informatie over de hardware en softwareomgeving van een (machine op een) omgeving. Deze informatie wordt aangeboden door het besturingssysteem.

Er zijn softwarepakketten die deze systeeminformatie (met een interval) opvragen om deze bijvoorbeeld te visualiseren of op te slaan voor analyse. Tijdens het onderzoeken naar beschikbare softwarepakketten die logbestanden monitoren kwam het Collectd protocol na voren. Dit is een standaard protocol dat beschrijft hoe systeeminformatie via het netwerk beschikbaar kan worden gesteld. Collectd is een opensource softwarepakket voor het Linux platform dat hardware informatie opvraagt en vervolgens via het netwerk beschikbaar stelt middels het Collectd protocol *(Collectd, z.d.)*.

Voor het Windowsplatform is er een implementatie van het Collectd protocol, genaamd SSC Server. Het nadeel aan SSC Server is echter dat de gratis versie de hardware informatie niet kan aanbieden met een interval onder de vijf minuten. De hardware informatie verkregen door SSC Server wordt opgevraagd via de Windows API *(SSC Server, z.d.)*. Binnen het .NET framework is het mogelijk om dezelfde Windows API te gebruiken om hardware informatie bijvoorbeeld naar een logbestand weg te schrijven.

4.1.3.1 Windows API .NET

In het .NET framework kan systeeminformatie opgevraagd worden met de System.Environment-, System.IO.DriveInfo klassen en System.Management library *(Naseri, 2013)*.

Door middel van de System.Environment klassen kan onder andere de Windows versie, de naam van het systeem, het pad naar de systeemmap en andere informatie over de huidige omgeving en het platform opgevraagd worden (Microsoft Developer Network, z.d.).

Door middel van de System.IO.DriveInfo klassen kan onder andere het volume label, de totale grootte (in bytes), de beschikbare ruimte (in bytes) en andere informatie over een harde schijf opgevraagd worden. De GetDrives functie van deze klassen retourneert een lijst van alle beschikbare logische harde schijven van het systeem *(Microsoft Developer Network, z.d.)*.

Door middel van de System.Management library kunnen alle andere beschikbare systeeminformatie opgevraagd worden die niet door de System.Environment- en System.IO.DriveInfo klassen verstrekt worden. Met de ManagementClass klassen kan onder andere de huidige processor -, geheugenbelasting, uitgevoerde processen en andere veranderlijke informatie opgevraagd worden. Deze informatie wordt door Windows Management Instrumentation (WMI) service verstrekt en door de ManagementClass ingelezen in een Win32 klassen.

In de officiële documentatie van het .NET framework is een lijst terug te vinden van alle beschikbare Win32 klassen. Deze zijn verdeeld in 6 categorieën die te zien zijn in *Tabel 4 - Overzicht WMI categorieën*.

Tabel 4 - Overzicht WMI categorieën

Categorie	Omschrijving
Computer System Hardware Classes	Hardware gerelateerde informatie.
Installed Applications Classes	Software gerelateerde informatie.
Operating System Classes	Besturingssysteem gerelateerde informatie
Performance Counter Classes	Prestatie gerelateerde informatie verstrekt door performance counters.
Security Descriptor Helper Class	Informatie omtrent beveilig en rechten op een systeem
WMI Service Management Classes	Informatie omtrent de WMI service.

4.1.4 Beschikbare software

Voor het in de gaten houden (monitoring) en beheren van logbestanden zijn er diversen soort softwarepakketten beschikbaar die zichzelf “log management tools” oftewel logbestandmanagers noemen. Het gezamenlijke doel van deze softwarepakketten is het centraal verzamelen en beheren van logbestanden. Om dit doel te kunnen verwezenlijken, bestaan deze logbestandmanagers uit twee applicaties: een manager en collector. Sommige pakketten hebben geen eigen collector en maken dan gebruik van collectorapplicaties van derden.

De collector is een (kleine) applicatie die geïnstalleerd wordt op het systeem dat gemonitord moet worden. Deze applicatie draait continu op de achtergrond en verstuurt de inhoud van logbestanden en/of systeeminformatie naar een centrale plek, de manager.

De manager is een serverapplicatie die van één of meerdere collectors gegevens verzamelt. De verdere invulling van de manager applicatie verschilt per softwarepakket. Enkele functionaliteiten die managers kunnen hebben is het filteren, bufferen, beheren en doorsturen van logbestanden en systeeminformatie.

In dit onderzoek zijn de functionaliteiten van een aantal van deze softwarepakketten kort beschreven en vergeleken met elkaar. De gekozen softwarepakketten zijn allemaal schaalbaar.

4.1.4.1 Fluentd

Figuur 2 - Fluentd (Fluentd, z.d.)

Fluentd is een opensource softwarepakket dat het Linux en Unix platform ondersteund. Voor het Windows platform is er een experimentele collectorapplicatie beschikbaar, maar is de manager applicatie nog in ontwikkeling. Het is echter mogelijk de collectorapplicatie voor het Windows platform te laten communiceren met de manager applicatie voor het Linux platform. Fluentd verzamelt en filtert gegevens uit logbestanden van diversen bronnen. Op hun website staat een uitgebreide lijst van welke bronnen (applicaties) ze allemaal ondersteunen.

De gegevens uit logbestanden die Fluentd ontvangt, hebben allemaal een andere structuur en worden door Fluentd getransformeerd in een standaard formaat, JSON. De gestandaardiseerde gegevens kunnen vervolgens door middel van criteria gefilterd worden. De gefilterde gegevens kunnen naderhand naar diverse systemen worden doorgestuurd voor notificatie, analyse of archivering.

Fluentd is geschreven in de programmeertaal C, in combinatie met Ruby en claimt weinig systeembronnen nodig te hebben. Naast een (online) community is er ook een commerciële ondersteuning mogelijk. Fluentd wordt door een aantal grote bedrijven gebruikt zoals bijvoorbeeld Amazon Web Services.

4.1.4.2 Graylog

Graylog is een opensource softwarepakket dat het Linux en Unix platform ondersteund. In tegenstelling tot Fluentd is er voor het Windowsplatform geen experimentele, maar doorontwikkelde collectorapplicatie beschikbaar.

Figuur 3 - Graylog (Graylog, z.d.)

Na het verzamelen van gegevens uit logbestanden, converteert Graylog de gegevens naar het Graylog Extended Log Format (GELF). Dit is een eigen ontwikkelde standaard om de gegevens te structureren.

Naast verzamelen en filteren, beheert en analyseert Graylog ook gegevens. Hiervoor worden Elasticsearch en MongoDB gebruikt. Graylog biedt een web interface aan waarmee gegevens gevisualiseerd of doorzocht kunnen worden, maar waar ook notificaties instelbaar zijn. Hoewel Graylog veel functionaliteiten bied is het mogelijk gegevens door te sturen naar enkele andere systemen waaronder Splunk (deze wordt later in dit onderzoek behandeld). Daarnaast heeft Graylog een eigen API die aangesproken kan worden. Ook is er een (online) community beschikbaar.

4.1.4.3 Logstash

Figuur 4 - Fluentd (Fluentd, z.d.)

Logstash is een crossplatform opensource softwarepakket waarmee gegevens uit logbestanden en metrieken verzameld kunnen worden. Logstash is een uit plug-ins opgebouwd systeem, zodat het gemakkelijk aangepast kan worden om zoveel mogelijk vormen van gegevens te kunnen verzamelen en verwerken. Door middel van input plug-ins is het naast logbestanden van applicaties ook mogelijk gegevens uit mailfolders en Twitter streams te gebruiken.

Naast verzamelen kan met Logstash de informatie ook gefilterd en aangepast worden. Hiervoor zijn filter plug-ins beschikbaar waarmee de

structuur van gegevens geëxtraheerd of verrijkt kan worden. Het is bijvoorbeeld mogelijk om in weblogs de geografische locatie van een bezoeker toe te voegen op basis van zijn IP adres.

De gefilterde en aangepaste gegevens kunnen vervolgens getransformeerd worden naar een standaard formaat, zoals JSON. Door middel van output plug-ins kunnen gegevens verstuurd worden naar diversen systemen voor notificatie, analyse, archivering of visualisatie. Daarnaast is het mogelijk gegevens naar een websocket of via HTTP naar een webservice te sturen. Hierdoor is het mogelijk om zelf een implementatie te maken van hoe gegevens behandeld moeten worden.

Logstash is ontwikkeld door Elastic en is onderdeel van de ELK (Elasticsearch, Logstash en Kibana) stack (verzameling applicaties dat een kant en klare oplossing biedt). Deze kant en klare oplossing biedt naast het verzamelen van gegevens ook het opslaan, doorzoeken en visualiseren van gegevens. Er is een grote (online) community en commerciële ondersteuning mogelijk.

4.1.4.4 Logentries

Logentries is een betaalde online dienst die gegevens uit logbestanden verzamelt, bewaard en analyseert. Ze werken samen met grote platformen waaronder Amazon Web Services, Microsoft Azure, Docker en Google Cloud. Afhankelijk van de gekozen licentie kan er naast de standaard functionaliteiten een aantal interessante analyses en notificaties gebruikt worden.

Figuur 5 - Logentries (Logentries logo, z.d.)

Met Anomaly Detection worden metrieken zoals bijvoorbeeld de response tijd van een webserver gebruikt om te bepalen of een applicatie nog na behoren functioneert. Deze metrieke worden ten op zichten van het gemiddelde bekeken en bij een te grote (instelbare) afwijking wordt een notificatie verstuurd.

Met Inactivity Alerting wordt de beschikbaarheid van applicaties in de gaten gehouden. Wanneer er voor een bepaalde tijd geen gegevens meer zijn ontvangen van een applicatie, wordt een notificatie verstuurd.

Door middel van een online dashboard kunnen alle instellingen gewijzigd worden en kunnen gegevens gevisualiseerd worden. Er is commerciële ondersteuning mogelijk.

4.1.4.5 Splunk

Splunk is een betaald crossplatform softwarepakket waarmee iedere vorm van machine gegenereerde data verwerkt en geanalyseerd kan worden. Er is van Splunk een Enterprise en een Light variant die beide on-premises (op de locatie van de klant geïmplementeerd) en als een dienst verkrijgbaar zijn.

Figuur 6 - Splunk (Splunk logo, z.d.)

Van alle onderzochten softwarepakketten bevat Splunk de meeste functionaliteiten. Naast de functionaliteiten die de eerder beschreven softwarepakketten bevatten, is het bijvoorbeeld mogelijk geautomatiseerd patronen te ontdekken in gegevens en deze periodiek te rapporteren. Naast analyseren op basis van patronen, worden er ook correlaties tussen gegevens gezocht.

Ondanks dat Splunk een behoorlijk compleet product is, zijn er diverse applicaties en plug-ins te verkrijgen op hun online portal genaamd Spunkbase. Bijvoorbeeld Microsoft SQL Server App, dit is een plug-in geschreven voor Microsoft SQL Server die de staat van de applicatie verstuurd naar Splunk. Er is commerciële ondersteuning mogelijk.

4.1.5 Conclusie

Om de doelstelling van dit onderzoek te behalen, is het noodzakelijk dat (de inhoud van) logbestanden en systeeminformatie verkregen en gecentraliseerd kan worden. De filter functionaliteiten die de verschillende softwarepakketten bieden is mooi meegenomen, maar alle andere functionaliteit is overbodig.

Het concept van sommige overbodige functionaliteiten is wel bruikbaar in het project. Anomaly Detection en Inactivity Alerting van Logentries is een interessante analyse die ook toegepast zou kunnen worden *(Logentries, z.d.)*.

Tabel 5 – Vergelijking softwarepakketten

Softwarepakket	Verkrijgen van (de inhoud van) logbestanden	Verkrijgen van systeeminformatie	Centraliseren van gegevens	Gratis	Lokaal beschikbaar
Fluentd	✓	✓	✓	✓	✓
Graylog	✓	✓	✓	✓	✓
Logstash	✓	✓	✓	✓	✓
Logentries	✓	✓	✓	✗	✗
Splunk	✓	✓	✓	✗	✓

Aangezien het om privacy gevoelige gegevens gaat is het van belang dat gegevens lokaal opgeslagen worden. De betaalde softwarepakketten bieden dezelfde voor dit onderzoek benodigde functionaliteiten en vallen daarom af. Van de overgebleven pakketten bevat Graylog de meeste overbodige functionaliteiten. Door een groter aantal overbodige functionaliteiten is de kans groot dat Graylog langzamer presteert dan Fluentd en Logstash.

De functionaliteiten en werking van Fluentd en Logstash is ongeveer het zelfde. Beide pakketten houden in de gaten of (de inhoud van) een logbestand veranderd is. Als een logbestand veranderd is, worden alleen de toevoegingen gelezen. Wanneer er een nieuw logbestand toegevoegd is, lezen beide pakketten de inhoud van begin tot einde.

Hoewel Fluentd en Logstash beide toepasbaar zijn, claimt Fluentd weinig systeembronnen nodig te hebben. Dit is aanneembaar omdat Fluentd in tegen stelling tot Logstash geen Java Virtual Machine (JVM) nodig heeft. Het geheugen gebruik van Logstash zal door de JVM hoger zijn dan Fluentd die per platform een zo compact mogelijke collector applicatie implementeert. Wanneer dit belangrijk is zal Fluentd de beste oplossing zijn.

Logstash wordt geadviseerd te gebruiken omdat, de ondersteuning voor Windows verder ontwikkeld en bewezen is. Daarnaast is Logstash flexibeler doordat het systeem opgebouwd is uit plug-ins. Hierdoor is het niet alleen toepasbaar in de huidige situatie maar wellicht ook in toekomstige situaties. Door de grote community is er een uitgebreide aanbod van plug-ins.

Ondanks dat er bestaande applicaties zijn die systeeminformatie aanbieden zoals Collectd en SSC Server is het interessant een eigen applicatie te ontwikkelen die systeeminformatie opvraagt en centraliseert. SSC Server heeft zijn beperkingen en met behulp van de klassen die het .NET framework beschikbaar stelt kan een applicatie ontwikkeld worden die met een kleinere interval, meer of specifiekere informatie kan aanbieden.

4.2 Centrale opslag

De verkregen gegevens zullen centraal opgeslagen worden in een NoSQL oplossing. Bij het beantwoorden van deze deelvraag is er door middel van literatuuronderzoek onderzocht welke NoSQL oplossingen geschikt zijn voor de opslag van de verkregen gegevens.

Bij het beantwoorden van deelvragen worden enkele andere vragen gesteld en beantwoord. Wat zijn NoSQL en welke NoSQL oplossingen worden veel gebruikt? Welke oplossingen kunnen gebruikt worden in combinatie met de softwarepakketten uit de vorige deelvraag?

In dit hoofdstuk staat de deelvraag "*Welke NoSQL oplossing is geschikt voor de centrale opslag van de gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie?"* centraal.

4.2.1 NoSQL

4.2.1.1 Algemeen

NoSQL is een term wat staat voor "Not only SQL" en wordt gebruikt als (verzamel)naam voor niet relationele databasesystemen. In de jaren 90 werden de beperkingen van relationele databasesystemen ontdekt, omdat er behoefte was om meer data en gebruikersaantallen aan te kunnen. De beperkingen konden niet opgelost worden binnen de bestaande niet relationele databasesystemen, met als gevolg dat er ander soort databasesystemen werden ontwikkeld *(Khan Zaki, 2014)*.

Bij relationele databasesystemen worden verzoeken verwerkt als transacties, zodat de integriteit van de data gewaarborgd blijft. ACID (Atomicity Consistent Isolation Durability) beschrijft hoe de integriteit van transacties gewaarborgd blijft *(Akhter Hossain & Moniruzzaman, 2013)*.

Een veel gebruikt voorbeeld waarom de ACID eigenschappen belangrijk zijn, is bij het verwerken van financiële gegevens. Wanneer het saldo van een rekening door een transactie aangepast wordt, is het belangrijk dat deze correct blijft. Zonder de ACID eigenschappen kan niet gegarandeerd worden dat een saldo niet overschreven wordt door een andere transactie die op hetzelfde moment plaatsvond. Om dit te verwezenlijken, wordt op bestandsniveau data gedurende een transactie "op slot" gezet door het databasesysteem, zodat deze niet overschreven kunnen worden. Het nadeel hiervan is dat gebruikers "in de wacht" staan zolang de data "op slot" is. Hierdoor ontstaan problemen omtrent de schaalbaarheid van dit type databasesystemen.

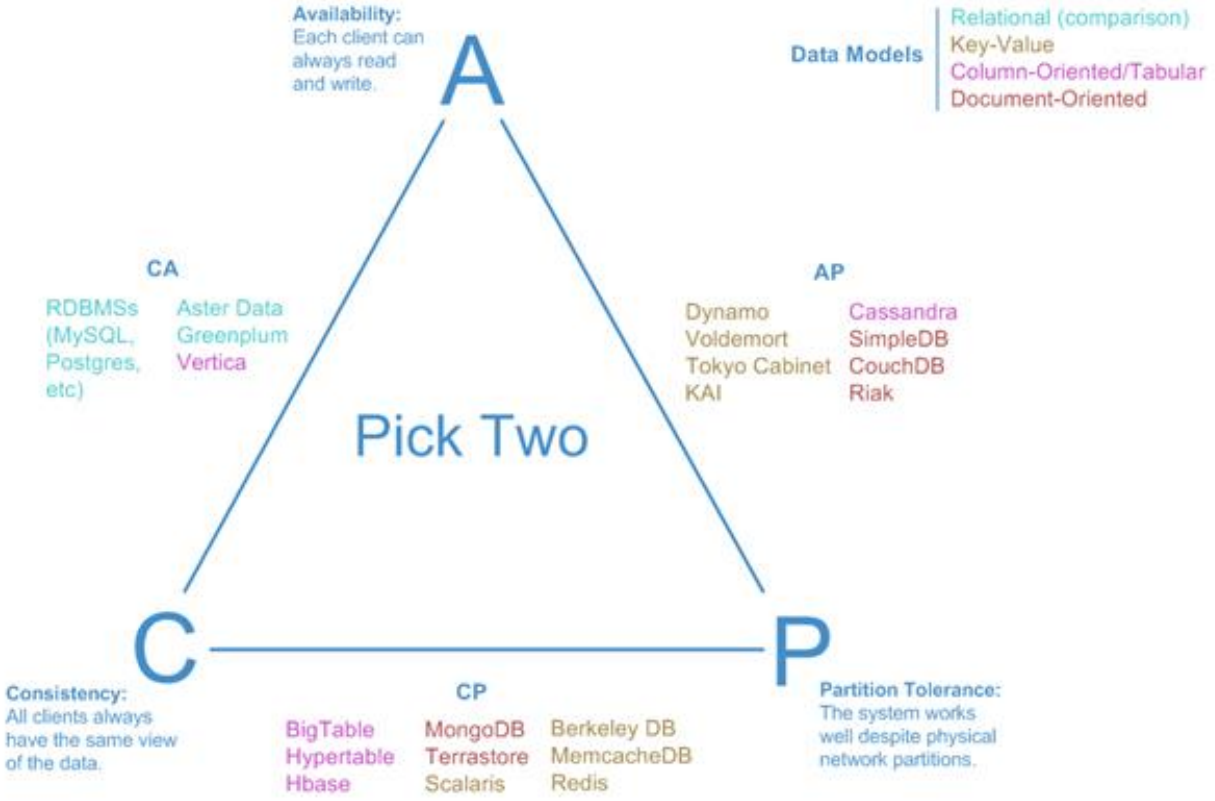
Bij niet relationele databasesystemen ligt de focus op schaalbaarheid en beschikbaarheid en is de consistentie van data vaak minder belangrijk. Deze kenmerken worden beschreven als BASE (Basically Available, Soft state, Eventually consistent), waarbij per NoSQL oplossing de verdeling tussen schaalbaarheid (Partition Tolerance), beschikbaarheid (Availability) en consistentie (Consistency) verschilt.

Met consistentie wordt bedoeld dat er altijd de meest recente gegevens worden aangeboden. Bij wijzingen worden gegevens als eerste aangepast voordat ze weer beschikbaar gesteld worden. Door middel van het consistentieniveau garandeer je de recentheid van gegevens.

Met beschikbaarheid wordt het altijd kunnen aan bieden van gegevens bedoeld. Het is van belang dat het systeem altijd beschikbaar is en in zo kort mogelijke tijd zoveel mogelijk aanvragen kan verwerken (ongeacht de recentheid van gegevens). Wanneer delen van het systeem niet meer functioneren, wordt er geprobeerd zo beschikbaar mogelijk te blijven.

Met Partition Tolerance wordt de indirecte schaalbaarheid van het systeem bedoeld. Het systeem als geheel moet te allen tijde blijven functioneren, ook als er geen communicatie mogelijk is tussen onderdelen. Om dit te bereiken moeten alle onderdelen afzonderlijke van elkaar kunnen functioneren.

Volgens de CAP stelling is het echter niet mogelijk om zowel schaalbaar, beschikbaar en consistent te zijn. Er moet altijd gekozen worden voor een combinatie van twee van de drie kenmerken (*Gilbert & Lynch*, 2002). In ***Error! eference source not found.***, zijn de drie kenmerken en de positionering van verschillende NoSQL oplossingen te zien.



Figuur 7 - Consistency, Availability en Partition Tolerance driehoek (Hurst, 2010)

Het is echter niet zo zwart-wit als de stelling suggereert. Wanneer een systeem “Availability-Partition Tolerance” is zoals bijvoorbeeld Cassandra, is de beschikbaarheid en schaalbaarheid van het systeem het belangrijkste, maar sluit dit niet uit dat de meest recente gegevens worden aangeboden. Gegevens worden uiteindelijk ook consistent gemaakt, alleen heeft dit lagere prioriteit.

In de meeste NoSQL oplossingen is het mogelijk de verdeling tussen de kenmerken in te stellen. Zo is het in Cassandra mogelijk verschillende consistentieniveaus in te stellen. Hiermee kan je garanderen dat gegevens niet op één, maar op bijvoorbeeld twee of zelfs alle systemen bijgewerkt worden *(Datastax, z.d.)*.

4.2.2 Classificatie

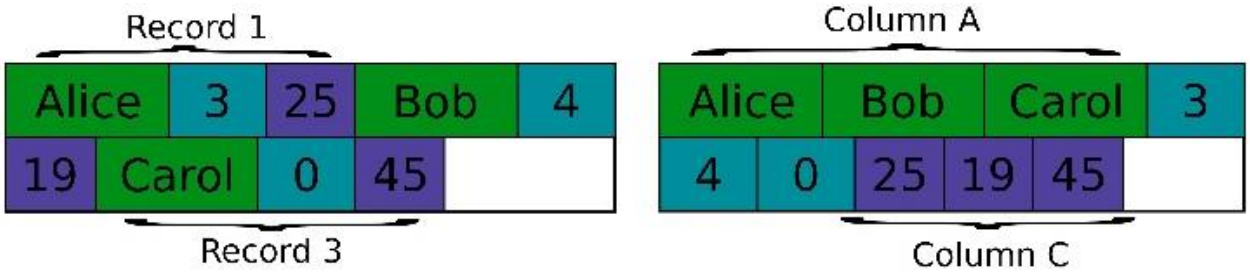
Op basis van de manier waarop de data opgeslagen wordt, kunnen NoSQL databasesystemen onderverdeeld worden in: key-value, (wide) column, document en graph databasesystemen.

4.2.2.1 Key-Value

Een simpele datastructuur waarbij, gegevens in paren worden opgeslagen en waarbij de sleutelnaam maar één keer voorkomt. Een paar bestaat uit een sleutelnaam (key) en een bijbehorend attribuut (value). Binnen deze datastructuur kunnen gegevens alleen via de sleutel opgehaald worden. Een bekend systeem dat gebruik maakt van Key-Value storage is het winkelwagensysteem van Amazon *(DeCandia, et al., 2007)*.

4.2.2.2 (Wide) Column

Een gedistribueerde kolom georiënteerde datastructuur. Gegevens worden in deze datastructuur niet op rij, maar op kolombasis bewaard. In ***Error! Reference source not found.*** is het verschil te zien, waarbij links de opslag op rijbasis n rechts de opslag op kolombasis is weergeven. Door gegevens op kolombasis te bewaren, wordt op fysiek (data) niveau alle kolomwaardes achter elkaar opgeslagen. Hierdoor kunnen specifieke kolommen doorzocht worden zonder overbodige data te lezen. Op rijbasis is dit onmogelijk omdat alle regels achter elkaar opgeslagen worden. Dit betekent dat alle kollommen die overbodig zijn, in iedere regels staan en dus gelezen worden. Het nadeel hieraan is dat de resultaten uit verschillende kollommen uiteindelijk samengevoegd moeten worden. Vanwege deze manier van opslag is het ook mogelijk verschillende aantal kollommen te gebruiken per rij.



Figuur 8 - (Wide) Column (Lipcon, 2009)

4.2.2.3 Document

Een document georiënteerd, dynamische datastructuur waarbij alle gegevens als XML, JSON (Javascript Option Notation) of BSON (Binary JSON) documenten worden opgeslagen. Ieder document kan een andere structuur bevatten en kan gevonden worden op basis van zijn inhoud. Databasesystemen die deze datastructuur gebruiken zijn over het algemeen goed in het inhoudelijk doorzoeken van tekstuele gegevens.

4.2.2.4 Graph

Een opslag voor gegevens waarbij de relaties tussen gegevens belangrijk zijn. Dit is de enige NoSQL variant die relaties bevat en wordt gebruikt bij bijvoorbeeld Social Media systemen. Gegevens in deze datastructuur (objecten) zijn opgeslagen in een grote verzameling zoals key-value pairs met daarnaast een verzameling relaties (edges) tussen alle objecten.

4.2.3 Databasesystemen

In dit onderzoek is voor iedere classificatie methode (behalve de graph classificatie), het meest gebruikte databasesysteem toegelicht *(DB-Engines, 2015)*.

4.2.3.1 Redis

Redis is een key-value georiënteerd opensource databasesysteem. De gegevens in deze databasesysteem worden in-memory bewaard. In plaats van een harde schijf, wordt het RAM geheugen gebruikt om gegevens in op te slaan. Het grote voordeel hiervan is de hogere verwerkingssnelheid van het RAM geheugen. Het grote nadeel hiervan is dat de opslag veel duurder is en dat er niet meer data dan RAM geheugen opgeslagen kan worden. Redis kan bij het vol lopen van de RAM-geheugen ruimte de minder frequent gebruikte gegevens verplaatsen naar de hardeschijf. Er zal op dat moment echter een dip ontstaan in de verwerkingssnelheid van Redis.

Gegevens worden in Redis als paren opgeslagen. De sleutelnaam van ieder paar wordt opgeslagen als een binairy-safe string. De bijbehorend attributen kunnen in een van de zeven verschillende soorten datastructuren worden opgeslagen *(Redis, z.d.)*. Van deze zeven soorten zijn de volgende vijf relevant:

- Binary-safe strings, characters, getallen of binaire gegevens tot 512 MB.
- Lists, verzameling string elementen gesorteerd op toevoeging. Geïmplementeerd als linked list waarbij het mogelijk is gegevens te zoeken of uit te breiden via de voor of achterkant van de verzameling.
- Sets, verzameling ongesorteerde unieke string elementen. Toevoegen, bewerken, verwijderen en zoeken (op sleutel) gaat op basis van hashes en is daarom van de orde 1.
- Sorted Sets, verzameling gesorteerde unieke string elementen. Aan ieder element wordt een floating point waarde toegekend, dit heet de score.
- Hashes, op basis van een sleutel kan hier een object in bewaard worden wat bestaan uit string naam waarde combinaties.

Bitmaps en hyperloglogs zijn de overgebleven twee soorten datastructuren die beide gebaseerd zijn op de Binary-safe strings datastructuur.

Het is mogelijk een vervaldatum (retentie) te koppelen aan gegevens paren. Wanneer de datum verstreken is worden de gegevens verwijderd. In Redis is het ook mogelijk kanalen te maken waarmee gegevens uit gewisseld kunnen worden door middel van het publish en subscribe patroon.

4.2.3.2 Cassandra

Cassandra is een (wide) column georiënteerd opensource databasesysteem. In Cassandra zijn meerdere Cassandra instanties (nodes) met elkaar verbonden zonder master-slave concept in een ring gebaseerde architectuur. Alle nodes zijn gelijk aan elkaar en communiceren door middel van een gedistribueerd en schaalbaar protocol, genaamd gossip (rondelen). Omdat iedere node gelijk aan elkaar is, is er geen “single point of failure” en stijgt of daalt de capaciteit van de totale ring lineaire bij iedere toegevoegde of verloren node.

Wanneer een node gegevens ontvangt worden deze als eerst op de harde schijf (in de commit log) opgeslagen. De commit log is een ongestructureerde lijst van ontvangen gegevens die het mogelijk maakt onverwerkte gegevens terug te halen. Vervolgens worden deze in het geheugen (in de memtable) opgeslagen. De memtable is een tijdelijke opslag waarbij data per column family (vergelijkbaar met een tabel) opgeslagen wordt tot een ingestelde grootte. Wanneer de ingestelde grote overschreven wordt, zullen alle gegevens in één keer worden verplaatst van het geheugen naar de harde schijf. Wanneer dit mislukt of als de node uitvalt kan de commit log gebruikt worden om de gegevens terug te halen.

4.2.3.3 MongoDB

MongoDB is een document georiënteerd opensource databasesysteem. De gegevens worden als een flexibiel document opgeslagen dat vergelijkbaar is met JSON. Ieder document kan wat betreft structuur van elkaar verschillen en bestaat uit één of meerdere velden. Voor veel gebruikte programmeertalen zijn er uitgebreide drivers beschikbaar die het mogelijk maken gegevens als objecten aan te spreken. Hierdoor hoeft er geen ORM laag geïmplementeerd worden. MongoDB is horizontaal schaalbaar. Dit betekend dat het mogelijk is de gegevens te verdelen over meerdere MongoDB servers.

Een van de onderdelen waar MongoDB goed in zijn en die interessant zijn voor de doelstelling van dit onderzoek, is het realtime analyseren van gegevens. Omdat MongoDB de inhoud van documenten kan doorzoeken, is het mogelijk analyses uit te voeren op databasesysteem niveau. Dit heeft als voordeel dat overbodige gegevens al op databasesysteem niveau kunnen worden gefilterd. Een databasesystemen zijn hier efficiënter in (bijvoorbeeld door gebruik van indexen) en de hoeveelheid gegevens die moet worden verstuurd over het netwerk is dan kleiner.

4.2.4 Gegevens

De gegevens uit logbestanden, systeem-, beschikbaarheids- en reactie informatie zijn onder te verdelen in twee categorieën. De logbestanden bevatten teksten in verschillende formaten die op verschillende momenten worden ontvangen. De hoeveelheid gegevens die ontvangen wordt is slecht in te schatten en kan verschillen per tijdseenheid. De overige gegevens zijn meetwaardes gerepresenteerd door een decimaal getal. Deze metingen worden op periodieke basis ontvangen. De hoeveelheid gegevens die ontvangen wordt is berekenbaar per tijdseenheid.

4.2.5 Logstash en Fluentd

Voor de centrale opslag van gegevens kan bij Logstash en Fluentd gebruik gemaakt worden van een aantal databasesystemen, waaronder Elasticsearch, Graphite, Influxdb, MongoDB en OpenTSD. Daarnaast is het bij Logstash nog mogelijk de gegevens naar een websocket of via http(S) naar een webservice te sturen. Het is zo mogelijk zelf een implementatie te maken van hoe gegevens opgeslagen moeten worden. De genoemde databasesystemen kunnen onderverdeeld worden in twee categorieën.

Elasticsearch en MongoDB zijn document georiënteerde NoSQL oplossingen. Elasticsearch is echter geen “echt” databasesysteem, maar een search engine ontwikkeld bovenop Apache Lucene. Gegevens in Apache Lucene worden als JSON documenten bewaard in een eigen opslagsysteem.

Graphite, InfluxDB en OpenTSDB zijn allemaal time series databasesystemen. Dit zijn databasesystemen die geoptimaliseerd zijn voor het bewaren van gegevens gekoppeld aan een tijd *(DB-Engines, z.d.)*. OpenTSDB is een systeem ontwikkeld bovenop HBase (een Column variant van NoSQL).

4.2.5.1 Time Series databasesystemen

Tijdens het onderzoek kwamen Time Series databasesystemen aan het licht. Dit zijn databasesystemen speciaal ontwikkeld voor het bewaren van gegevens gekoppeld aan tijd voor analyse of visualisatie doeleinden. In een Time Series databasesysteem zijn metingen altijd gekoppeld aan een tijd. Omdat gegevens op basis van tijd vaak na een bepaalde tijd niet meer relevant zijn, werken Time Series databasesystemen met een retentiebeleid. Dit houdt in dat bijvoorbeeld alle gegevens maar voor een vooraf ingestelde periode bewaard blijven. Gegevens worden vervolgens automatisch opgeruimd wanneer ze deze periode overstijgen.

4.2.5.2 InfluxDB

Uit een vergelijking van Graphite, InfluxDB en OpenTSDB blijkt dat Graphite alleen maar numerieke gegevens kan opslaan. Dit geld niet voor InfluxDB en OpenTSDB, die kunnen beiden zowel numerieke als tekst gegevens opslaan. InfluxDB heeft ten op zichten van OpenTSDB een grotere community.

Ondanks dat InfluxDB geen relationeel databasesysteem is, zijn een aantal aspecten bijna hetzelfde. In InfluxDB heb je geen tabellen, maar measurements. Zo'n measurement kan je een naam geven en bevat meerdere points. Points zijn meetwaardes gekoppeld aan een tijd. Daarnaast kan er gebruik gemaakt worden van tags (vergelijkbaar met geïndexeerde kolommen) en fields (vergelijkbaar met niet-geïndexeerde kolommen) om extra informatie te koppelen aan points. De gegevens in zo'n measurement worden tijdreeksen genoemd. Het is in InfluxDB mogelijk meerdere tijdreeksen tegelijk op te vragen. Deze tijdreeksen kunnen helaas niet aan elkaar gekoppeld worden. Er bestaat geen join statement zoals in relationele databasesystemen. Dit is het meest gebruikte Time Series databasesysteem dat voor zowel .NET als de programmeertaal R ondersteuning biedt *(DB-Engines, z.d.)*.

InfluxDB maakt onderliggend gebruik van de BoltDB storage engine *(InfluxDB, z.d.)*. BoltDB is een key-value storage.

4.2.6 Azure

Azure is het cloudcomputingplatform van Microsoft. Op dit platform kunnen verschillende diensten afgenomen worden, waaronder verschillende soorten databasesystemen. Deze diensten worden aangeboden in twee varianten: "Software as a Service" en als "Virtual Machine".

Met "Software as a Service" maak je als klant gebruik van een softwarepakket als dienst. Bij het afnemen van deze dienst heb je als bedrijf geen kapitaalinvestering nodig en betaal je maandelijks alleen voor het verbruik. Daarnaast zorgt Azure voor het beheer, onderhoud en ondersteuning van deze dienst. Je hebt als klant alleen toegang tot de afgenomen diensten. Deze dienstvariant wordt ook wel eens Software on Demand genoemd.

Met "Virtual Machine" krijg je, zoals de naam al zegt, toegang tot een virtuele machine. In tegenstelling tot "Software as a Service" ben je bij deze dienstvariant als klant de beheerder van het systeem. Door zelf verantwoordelijk te zijn voor het beheer, zijn de mogelijkheden bij deze dienstvariant het grootst. Het nadeel is echter dat je zelf verantwoordelijk bent voor het systeem wat je afneemt en dat de ondersteuning beperkter zal zijn. Je hebt bijvoorbeeld geen ondersteuning op de software die je erop installeert.

Azure stelt de onderstaande NoSQL oplossing beschikbaar as a service:

- Key-Value, door middel van Tables.
- (Wide) Column, door middel van HBase.
- Document, door middel van DocumentDB

Voor deze drie databasessystemen zijn .NET connectoren beschikbaar. Graph databasesystemen worden niet ondersteund door Azure, omdat deze NoSQL variant nog niet veel gebruikt wordt.

Naast databasesystemen biedt Azure ook een dienst voor analytisch doeleinde, genaamd Azure Machine Learning. In deze dienst is het mogelijk data te analyseren om daar vervolgens voorspellende modellen van te maken. De gerealiseerde modellen kunnen vervolgens gepubliceerd worden, zodat ze toegankelijk zijn als REST API.

4.2.7 Conclusie

De gegevens uit logbestanden zijn anders en worden op een andere frequentie ontvangen dan gegevens uit systeem-, beschikbaarheids- en reactie informatie. Het advies gaat daarom uit de twee verschillende soorten gegevens op te slaan in twee verschillende soorten oplossingen.

De gegevens uit logbestanden bevatten teksten in verschillende formaten. Dit past goed bij document georiënteerde NoSQL oplossingen. Deze classificatie NoSQL oplossing is sterk in het bewaren en doorzoeken van verschillende formaten gegevens. DocumentDB en MongoDB zijn beide document georiënteerde NoSQL oplossingen die beschikbaar zijn in Azure. Standaard wordt alleen MongoDB door Fluentd en Logstash ondersteund. DocumentDB heeft echter de voorkeur omdat deze “as a service” wordt aangeboden. Dit heeft als voordeel dat Azure verantwoordelijk is voor beheer, onderhoud en ondersteuning van DocumentDB. Met Logstash is het mogelijk gegevens te versturen naar een eigen ontwikkelde web service, die de gegevens vervolgens bewaart in DocumentDB.

Voor de gegevens uit systeem-, beschikbaarheids- en reactie informatie kan het beste een Time Series databasesysteem gebruikt worden. Het zijn geen echte NoSQL oplossingen, maar uit dit onderzoek blijkt dat deze variant databasesystemen speciaal ontwikkeld is voor de opslag en voor het analysen van tijd gedreven gegevens. Omdat Time Series databasesystemen niet veel gebruikt worden, kan het gebruik hiervan een risico vormen. Dit risico is bij InfluxDB echter minimaal, omdat er een grote community en documentatie beschikbaar is. Ondanks het risico gaat het advies gaat naar InfluxDB. In Azure kan InfluxDB door middel van een virtual machine beschikbaar worden gesteld.

4.3 Afwijkend gedrag herkennen

Er wordt onderzocht of er aan de hand van afwijkend gedrag, voorspellingen over de toekomst gemaakt kunnen worden. Bij het beantwoorden van deze deelvraag is er door middel van literatuuronderzoek onderzocht op welke manier afwijkend gedrag herkend kan worden.

Bij het beantwoorden van deelvragen worden enkele andere vragen gesteld en beantwoord. Hoe analyseer en "leer" je van gegevens? Wat is afwijkend gedrag en hoe herken je het?

In dit hoofdstuk staat de deelvraag *"Hoe herken je afwijkend gedrag uit (de inhoud van) logbestanden, systeem-, beschikbaarheids- en reactie informatie?*" centraal.

4.3.1 Machine Learning

Arthur Samuel definieerde in 1959 machine learning als een studiegebied dat computers de mogelijkheid biedt te leren van gegevens zonder dat expliciet te programmeren *(Munoz, z.d.)*. De manier waarop dit bereikt kan worden, verschilt aan de hand van het probleem dat opgelost moet worden. De aanpak hiervan is onder te verdelen in supervised-, unsupervised- en reinforcement learning.

4.3.1.1 Supervised learning

Wanneer er van voorbeelden (input) het resultaat (output) bekend is, kan voor iedere nieuwe input een bijpassend resultaat voorspeld worden. Door een patroon te zoeken in de voorbeelden, kunnen door middel van een algoritme voorspellingen gemaakt worden. Een simpel voorbeeld hiervan is een lineaire voorspelling van gewicht op basis van lengte. Iemand die langer is, zal gemiddeld zwaarder zijn dan iemand die kleiner is. Wanneer de lengte en het gewicht van personen bekend is, kan een (gemiddeld) gewicht voor iedere lengte worden voorspeld.

4.3.1.2 Unsupervised learning

Wanneer er alleen voorbeelden (input) beschikbaar zijn en het resultaat (output) nog onbekend is. Door correlaties te zoeken en gegevens te groeperen, kunnen patronen ontdekt worden. Deze aanpak methode wordt ook gebruikt om afwijkend gedrag te detecteren. Ondanks het dat resultaat onbekend is, kunnen hier wel conclusies getrokken worden. Als een voorbeeld een afwijkend resultaat heeft van de rest van de voorbeelden, kan geconcludeerd worden dat dit afwijkend gedrag is. Door dit te classificeren als afwijkend gedrag kan van een nieuwe input voorspeld worden of dit een afwijkend resultaat gaat hebben of niet.

4.3.1.3 Reinforcement learning

Wanneer er gestreefd wordt naar een doel en de voorbeelden en resultaten hiervan nog onbekend zijn. Dit wordt veel gebruikt om bijvoorbeeld kunstmatige intelligentie te trainen. Deze aanpak werkt met een beloningssysteem die een beloning geeft voor ieder voorbeeld waarmee het doel wordt bereikt. Een voorbeeld hiervan is een robot leren hoe een pannenkoek gedraaid moet worden. Als de pannenkoek omgedraaid wordt en in de pan blijft, vindt er een positieve beloning plaats *(Lison, 2012)*. Wanneer de pannenkoek uit de pan valt of niet omgedraaid is, vindt er een negatieve of zeer negatieve beloning plaats. Naar mate er meer voorbeelden beschikbaar zijn en aan ieder voorbeeld een beloning hangt, is het duidelijk hoe het doel bereikt kan worden.

4.3.2 Afwijkend gedrag

Afwijkend gedrag kan verwacht en onverwacht plaatsvinden. Wanneer een systeem niet meer toegankelijk is, kan verwacht worden dat systemen die hier gebruik van maken, afwijkend gedrag gaan vertonen.

Dat een systeem ontoegankelijk wordt, kan herkend worden door middel van systeeminformatie. Wanneer een systeem overbelast wordt, kan dit resulteren in een ontoegankelijk systeem. Daarnaast kunnen gegevens uit de beschikbaarheids- en reactie informatie gebruikt worden om te herkennen of systemen niet meer toegankelijk zijn of langzamer reageren dan gewenst.

Uit (de inhoud van) logbestanden kan ook afwijkend gedrag herkend worden. Dit kan bijvoorbeeld herkend worden door tekst in logberichten of de Severity (bijvoorbeeld waarschuwing).

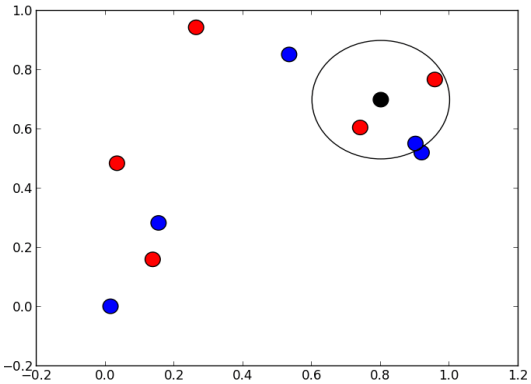
Je hebt ook onverwacht afwijkend gedrag dat ontstaat door software bugs. Wanneer een software bug optreedt, zullen excepties gegenereerd worden. Dit hoeft niet direct afwijkend gedrag te zijn, maar als het binnen een korte tijdspanne meerdere keren voorkomt, is het waarschijnlijk wel afwijkend gedrag.

4.3.3 Herkenning

Door middel van algoritmes kunnen gegevens geïnterpreteerd worden om patronen of informatie te ontdekken. De algoritmes die gebruikt worden in supervised- en unsupervised learning, zijn onder te verdelen in de categorieën classification, regression en clustering algorithms.

Time Series Analysis en Iterative Partitioning Log Mining zijn algoritmes die in meerdere onderzoeken genoemd worden en interessant zijn voor dit onderzoek.

4.3.3.1 Classification algorithms



Figuur 9 - K-nearest neighbors (ŷhat, 2013)

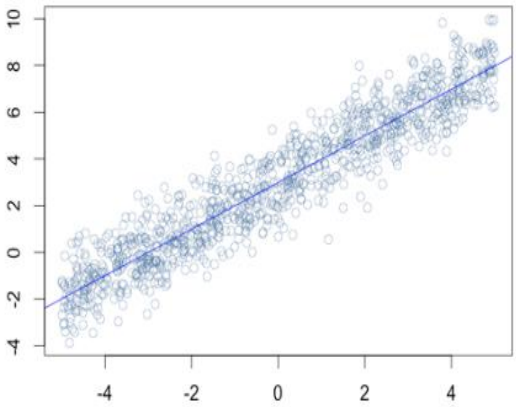
Dit zijn algoritmes die aan de hand van gelabelde gegevens, nieuwe gegevens classificeren. De beschikbare gegevens zijn voorbeelden waarvan voor ieder voorbeeld bekend is in welke vooraf gedefinieerde groep (classificering) deze hoort. Met deze categorie algoritmes kunnen problemen opgelost worden waarvoor het gewenste resultaat een classificering is.

Met classificering wordt bedoeld het toekennen van gegevens aan één van de twee of meer beschikbare categorieën. Voor ieder onbekend gegeven wil je bijvoorbeeld weten in welke groep deze hoort. Wat voor genre is dit nummer? Is dit een man of een vrouw?

Deze vorm algoritmes worden toegepast in supervised learning. Door van geclassificeerde gegevens te leren, wordt het mogelijk nieuwe gegevens te classificeren. Een veelgebruikt algoritme hiervoor is K-nearest-neighbors. In dit algoritme worden nieuwe gegevens geclassificeerd door te kijken naar omliggende gegevens (zie ***Error! Reference ource not found.***). Het aantal omliggende gegevens waarnaar gekeken wordt, is instelbaar en beïnvloedt de correctheid. In een bepaalde mate zorgen meer omliggende gegevens voor een beter antwoord. Wanneer het aantal echter te hoog is, kunnen gegevens verkeerd geclassificeerd worden omdat irrelevante gegevens meegenomen worden in de classificatiebepaling *(Fayyad, Piatetsky-Shapiro, & Smyth, 1996)*.

4.3.3.2 Regression algorithms

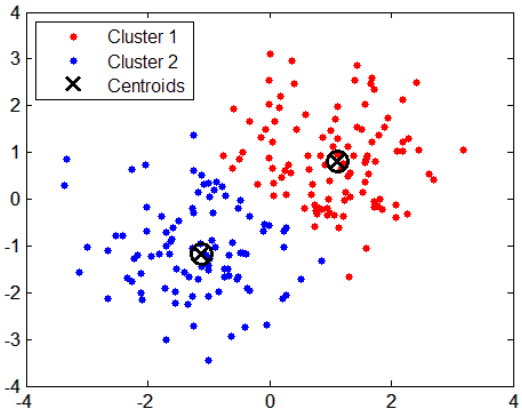
Dit zijn algoritmes die aan de hand van gegeven waardes (input), bijbehorende waardes (output) berekenen. Met deze categorie algoritmes kunnen problemen opgelost worden waarbij voor gegeven waardes een bijbehorende waarde berekend moet worden. Bijvoorbeeld als ik x kilometer per uur rij is mijn remweg y meter lang. Of andersom als mijn remweg y lang is reed ik x kilometer per uur.



Figuur 10 - Linear Regression (Bare, 2012)

Deze vorm algoritmes worden toegepast in supervised learning. Een veel gebruikt algoritme hiervoor is Linear regression zie **Error! eference source not found.**. In dit algoritme wordt aan de hand van gegeven waardes een lijn berekend die de waardes representeert. De manier waarop deze lijn door de gegevens loopt kan uitgedrukt worden in een twee getallen, de asafsnede (oorsprong) en de richtingscoëfficiënt. Door een gegeven waarde (input) te vermenigvuldigen met de richtingscoëfficiënt en op te tellen bij de asafsnede, kan bijbehorende waarde (output) berekend worden *(Wageningen Universiteit, 2013)*.

4.3.3.3 Clustering algorithms



Figuur 11 - K-means (Humanoriented, z.d.)

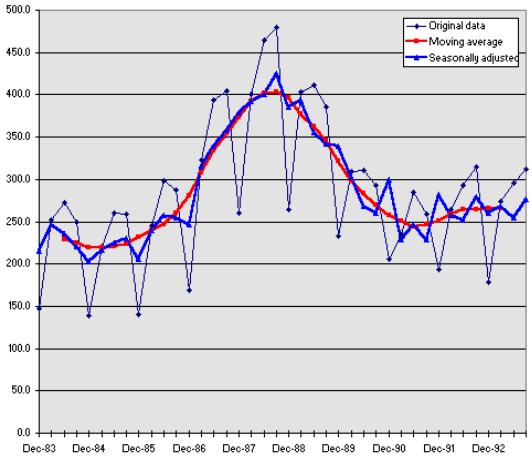
Dit zijn algoritmes die gegevens groeperen om patronen te vinden. Met deze categorie algoritmes worden problemen opgelost, waarbij het herkennen van een patroon het gewenste resultaat is.

Deze vorm van algoritmes worden toegepast in unsupervised learning. Een veel gebruikt algoritme dat gegevens groepeert heet K-means (zie ***Error! Reference source not found.***). In dit algoritme pecificeer je het aantal clusters (groepen) dat je wilt vinden in de gegevens. Het algoritme bereikt dit door te beginnen met voor het aantal gewenste clusters, random waardes te selecteren. Iedere random waarde wordt het centrum van een cluster. Vervolgens worden waardes iteratief toegekend aan het cluster met de kleinste afstand tot het (cluster) centrum. Bij iedere verandering van het cluster wordt een nieuw centrum bepaald. Omdat het centrum van een cluster veranderlijk is, kunnen waardes van cluster veranderen. Het algoritme stopt als alle waardes toegekend zijn aan een cluster en ieder (cluster) centrum onveranderlijk is.

4.3.3.4 Time Series Analysis

Time Series Analysis, een regressie algoritme zie ***Error! Reference ource not found.***, wordt gebruikt om tijd gedreven gegevens te analyseren. Door middel van het voortschrijdend gemiddelde (moving average) kan een trend worden herkend. Dit is een gemiddelde waarde van een vooraf bepaalde periode die verstreken is. Met deze moving average kan een afgevlakte lijn door gegevens berekend worden. De lengte van de periode bepaalt de afvlakking en afwijking van de lijn. Hoe vlakker de lijn, hoe groter de kans op afwijkingen.

Naast de trends die de moving average herkent, zijn er ook afwijkingen die herhaaldelijk over tijd ontstaan. Een voorbeeld hiervan is bijvoorbeeld de belasting van een systeem over tijd. Gedurende werktijden is de kans groot dat een systeem zwaarder belast wordt. Deze afwijking kan worden berekend en heet de seasonal adjustment. Door de gemiddelde belasting van ieder uur van de dag te vergelijken met de gemiddelde belasting van de dag, kan een correctie factor per uur berekend worden.



Figuur 12 - Time Series Analysis (Fuqua School of Business, z.d.)

De periode van de moving average en de seasonal adjustment kan, maar hoeft niet gelijk aan elkaar te zijn. Met een gemiddelde moving average en seasonal adjustment kan vervolgens een voorspelling over de toekomst gemaakt worden.

Time Series Analysis wordt veel gebruikt om een voorspelling te maken over de toekomst aan de hand van gegevens uit het verleden. Door middel van het moving average model kan bijvoorbeeld het processorgebruik voorspeld worden *(C. Waclawski, z.d.)*.

4.3.3.5 Iterative Partitioning Log Mining

Ook wel IPLoM genoemd. Dit is een algoritme dat speciaal ontwikkeld is voor het interpreteren van (de inhoud van) logbestanden. Door middel van dit algoritme worden logberichten door middel van vier stappen geanalyseerd om de verschillende soorten logberichtformaten te detecteren. Vervolgens kunnen de dynamische en constante woorden gescheiden worden, zodat de dynamische waardes gebruikt kunnen worden.

De eerste stap van dit algoritme is het sorteren van berichten op basis van het aantal woorden. Daarna wordt de frequentie dat een woord voortkomt gebruikt om te bepalen welke woorden constant zijn en welke dynamisch zijn. Waardes die veel voorkomen zijn waarschijnlijk constante waarden. Weinig voorkomende woorden zijn waarschijnlijk dynamisch. Vervolgens worden bijectie gezocht tussen woorden. Dit is een wiskundige term voor het zoeken van koppelingen tussen waardes in een verzameling. Tot slot kan aan de hand van het resultaat van de vorige drie stappen bepaald worden welke formaten logberichten er zijn.

Een voorbeeld van de inhoud van een logbestand is *(Makanju, Nur Zincir-Heywood, & Evangelos, 2009)*.

```
“      Connection from 192.168.10.6 port 25
       Connection from 192.168.10.6 port 80
       Connection from 192.168.10.7 port 25
       Connection from 192.168.10.8 port 21    ”
```

In dit voorbeeld bevatten alle berichten het zelfde aantal woorden. De woorden “Connection”, “from” en “port” komen het meeste voor. Met de eerste twee stappen van het algoritme kan al bepaald worden dat het logformaat waarschijnlijk “Connection from * port *” zal zijn, waarbij het * de dynamische waardes representeert *(Makanju, Nur Zincir-Heywood, & Evangelos, 2009)*.

Dit algoritme kan toegepast worden om dynamische waardes te extraheren uit (de inhoud van) logbestanden. Door nieuwe variabelen waardes te vergelijken met de variabelen waardes uit het verleden, kan afwijkend gedrag herkend worden.

Wanneer in een logbericht bijvoorbeeld de reactietijd van een externe databron wordt vermeld, kan deze reactietijd vergeleken worden met reactietijden uit het verleden. De reactietijd kan uit het logbericht geëxtraheerd worden door middel van het IPLoM algoritme.

IPLoM is een goed algoritme voor het verkrijgen van informatie uit de inhoud van logbestanden *(Makanju, Nur Zincir-Heywood, & Evangelos, 2009)*.

4.3.4 Conclusie

Verwacht afwijkend gedrag kan door middel van supervised learning herkend worden. Door middel van unsupervised learning kunnen patronen ontdekt worden in onverwacht afwijkend gedrag. Door het herkennen van patronen kan onverwacht afwijkend gedrag mogelijk herkend worden.

Classificatie algoritmes zijn het meest geschikt om te bepalen of iets afwijkend gedrag is of niet. Dit zijn namelijk algoritmes waarbij het antwoord een classificering is. Wanneer gegevens (gedrag) uit het verleden bekend zijn, wordt het mogelijk nieuwe gegevens te classificeren als verwacht of afwijkend gedrag.

Daarnaast kunnen door middel van regressie algoritmes de kans berekend worden dat een nieuwe gegeven afwijkend gedrag is berekend worden. Om classificatie of kansberekening uit te kunnen voeren, is het noodzakelijk dat er criteria worden opgesteld wanneer iets afwijkend gedrag is.

Door clustering algoritmes toe te passen op de inhoud van logbestanden wordt het mogelijk patronen te herkennen. Daarnaast kan geclusterde data handmatig geclassificeerd worden zodat het mogelijk wordt classificatie algoritmes toe te passen. Wanneer het gewenste resultaat niet met clustering wordt bereikt, kunnen door middel van het IPLoM algoritme dynamische waardes uit (de inhoud van) logbestanden worden geëxtraheerd.

Systeem-, beschikbaarheids- en reactie informatie kan ook door middel van regressie algoritmes zoals Time Series Analysis geanalyseerd worden. Door de systeem-, beschikbaarheids- en reactie informatie van het verleden te analyseren kunnen toekomstige waardes voorspeld worden. Wanneer een waarde niet gelijk is aan de voorspelling, is het afwijkend gedrag.

4.4 Voorspellen

Om aan de hand van afwijkend gedrag problemen te voorspellen moet de relatie tussen afwijkend gedrag en problemen bekend zijn. Bij het beantwoorden van deze deelvraag is er door middel van literatuuronderzoek onderzocht hoe problemen herkent en voorspeld kunnen worden.

Bij het beantwoorden van deelvragen worden enkele andere vragen gesteld en beantwoord. Welke problemen zijn er bijvoorbeeld? Welke relatie hebben problemen en afwijkend gedrag? Hoe kan je voorspellen maken?

In dit hoofdstuk staat de deelvraag "*Hoe kun je aan de hand van afwijkend gedrag problemen herkennen zodat deze voorspeld kunnen worden?*" centraal.

4.4.1 Problemen

Er zijn verschillende soort problemen die plaats vinden op systemen of applicaties. Net als afwijkend gedrag ontstaan problemen verwacht en onverwacht. Dit wil niet zeggen dat ieder verwacht of onverwacht afwijkend gedrag resulteert in een verwacht of onverwacht probleem.

Verwachte problemen zijn bijvoorbeeld het vol lopen van een hardeschijf, het ontoegankelijk worden van een systeem en het vast lopen van een applicatie. Het vol lopen van een hardeschijf kan gekoppeld worden aan zowel verwacht gedrag als verwacht afwijkend gedrag. Wanneer de hardeschijf iedere dag voor één dertigste gevuld wordt, kan verwacht worden dat deze na dertig dagen vol is. Het kan ook zijn dat het gebruik van de hardeschijf voor een periode hoger of lager is dan normaal. Dit is verwacht afwijkend gedrag dat is ontstaan uit een afwijking ten opzichte van de voorspelde belasting. Dit verwacht afwijkend gedrag zou kunnen resulteren in het ontstaan van een probleem. Er kunnen bijvoorbeeld geen gegevens meer naar de hardeschijf geschreven worden. Dit is een probleem. Het kan echter ook zijn dat die dag oude bestanden verwijderd werden en dat daarom het hardeschijf gebruik minder was. Dit is echter geen probleem.

Dit zelfde geld voor het ontoegankelijk raken van een systeem. Wanneer een systeem iedere woensdagmiddag een gemiddelde belasting heeft van bijvoorbeeld 70% is de verwachting dat de volgende woensdagmiddag dezelfde gemiddelde belasting zal plaats vinden. Het is afwijkend gedrag wanneer deze belasting hoger of lager is dan normaal. Een hogere belasting kan bijvoorbeeld resulteren in het ontoegankelijk worden van het systeem. Dit hoeft echter niet het geval te zijn. Wanneer bijvoorbeeld een minder frequente handeling zoals het installeren, bijwerken of verwijderen van software plaats vind op een systeem, is de belasting tijdelijk hoger. Deze tijdelijke belasting is echter geen probleem.

Onverwachte problemen zoals bijvoorbeeld het vast lopen van een applicatie kunnen gekoppeld worden aan onverwacht afwijkend gedrag. Wanneer een logbericht met de Severity “emergency / fatal” en “error” wordt

weggeschreven naar een logbestand kan het zijn dat het beschreven evenement heeft geresulteerd in het vast lopen van een applicatie.

Daarnaast zijn er problemen die onverwacht ontstaan en nog nooit eerder hebben plaats gevonden. Voor dit soort problemen is er op het moment van ontstaan nog geen gerelateerd afwijkend gedrag bekend.

4.4.2 Voorspellen

Het ontstaan van problemen uit verwacht afwijkend gedrag kan door middel van regressie algoritmes voorspeld worden. Door middel van gegevens uit het verleden kan berekend worden wanneer een probleem gaat ontstaan. Via lineaire regressie kan bijvoorbeeld de groei (richtingscoëfficiënt) en het startpunt (asafsnede) van hardeschijf gebruik herkend worden. De formule waarmee vervolgens hardeschijf gebruik (γ) in de toekomst voorspeld kan worden is als volgt:

$$\gamma = \beta_0 + \beta_1 \chi$$

Waarbij β_0 de waarde van de asafsnede, β_1 de waarde van de richtingscoëfficiënt en χ de tijd representeert. Wanneer een lege hardeschijf iedere dag voor één dertigste gevuld wordt. Is de schijfruimte na 3 dagen voor één tiende gevuld. Door de formule om te schrijven kan de tijd voorspeld worden. Deze formule is als volgt:

$$\chi = \frac{\gamma - \beta_0}{\beta_1}$$

Wanneer een voor de helft gevulde hardeschijf iedere dag voor één dertigste gevuld wordt, kan verwacht worden dat na 15 dagen de hardeschijf vol is.

Voor Time Series Analysis kan het zelfde gedaan worden. Aan de hand van gegevens uit het verleden kan een trend herkend worden. Door middel van de trend kunnen toekomstige waardes berekend worden. Wanneer de verwachte waarde voor een x tijd afwijkt kan verwacht worden dat een systeem overbelast en daardoor ontoegankelijk wordt. Dit probleem kan niet exact voorspeld worden en zal daarom door een instelbare tijdsperiode voorspeld worden. Wanneer een systeem bijvoorbeeld vijf minuten overbelast is, kan een systeem ontoegankelijk worden.

Het ontstaan van problemen uit onverwacht afwijkend gedrag kan moeilijk tot niet voorspeld worden. Het onverwacht afwijkend gedrag dat gerelateerd is aan het ontstaan van een probleem is namelijk onverwacht ontstaan. Wel is het mogelijk een voorspelling te maken aan de hand van de frequentie waarop logberichten ontstaan. Door (de inhoud van) logbestanden te analyseren en logberichten te groeperen per Severity kan er inzicht getoond worden in het aantal logberichten per Severity. Het aantal logberichten met de Severity “emergency / fatal”, “error” en “warning” kunnen vergeleken worden met het aantal logberichten met dezelfde Severity dat heeft plaats gevonden in het verleden. Wanneer het aantal hoger is dan normaal is de kans dat een onverwacht probleem ontstaat groter.

4.4.3 Profielen

Wanneer er een probleem is ontstaan op een systeem of applicatie, kan er worden verwacht dat andere systemen en applicaties die er afhankelijk van zijn ook problemen gaan krijgen. Door profielen van systemen en applicaties te maken en hun relaties te registeren kunnen dit soort problemen voorspeld worden.

In een profiel staat dan bijvoorbeeld applicatie X op systeem 1, maakt gebruik van applicatie Y op systeem 2. Wanneer er dan een probleem ontstaat bij applicatie Y kan voorspeld worden dat er mogelijk een probleem gaat ontstaan op applicatie X.

Daarnaast kunnen alle applicatie Y instanties van verschillende omgevingen met elkaar vergeleken worden. Als op een bepaalde instantie een probleem herhalend plaats vind, kan voorspeld worden dat andere instanties dit mogelijk ook krijgen.

Door instanties te vergelijken kan er worden herkent of een instantie afwijkend gedrag vertoont. Wanneer een instantie afwijkend gedrag vertoont kan voorspeld worden dat een probleem kan ontstaan. Dit is echter een theoretische mogelijkheid omdat de omstandigheden van omgevingen vaak anders zijn. Omgevingen met hogere capaciteit zullen anders reageren dan omgevingen met een lagere capaciteit. Daarnaast verschilt het aantal applicaties dat aangeboden wordt tussen omgevingen.

Naast bovengenoemde relaties maken veel omgevingen gebruik van de zelfde fysieke opslag. Deze relatie tussen omgevingen kan ook gebruikt worden om problemen te voorspellen. Wanneer de opslagcapaciteit van een x aantal omgevingen voor een x tijd overbelast worden, kan voorspeld worden dat andere omgevingen op dezelfde fysieke opslag mogelijk problemen gaan krijgen.

4.4.4 Conclusie

Het ontstaan van problemen kan in veel situaties gekoppeld worden aan afwijkend gedrag. De problemen die gekoppeld kunnen worden aan verwacht afwijkend gedrag, kunnen voorspeld worden. Door verwacht afwijkend gedrag te herkennen voordat het plaats vind, is het dus mogelijk te voorspellen dat problemen gekoppeld aan het betreffende afwijkend gedrag mogelijk gaat plaats vinden. Voor verwacht afwijkend gedrag zoals bijvoorbeeld het vol lopen van een hardeschijf kan zelfs het moment waarop het afwijkend gedrag gaat plaats vinden berekend worden.

Problemen die gekoppeld zijn aan onverwacht afwijkend gedrag kunnen moeilijk tot niet voorspeld worden. Wanneer onverwacht afwijkend gedrag is gedetecteerd kan het zijn dat een probleem al heeft plaats gevonden. Dat betekend dat dit soort problemen niet voorspeld kunnen worden. Het is echter wel mogelijk de kans dat een onverwachts probleem gaat ontstaan te beredeneren aan de hand van (de inhoud van) logbestanden. Door Time Series Analysis toe te passen op het aantal logberichten dat ontstaat per x tijd kan een voorspelling gemaakt worden.

Problemen die niet gekoppeld zijn aan afwijkend gedrag kunnen niet voorspeld worden.

5. Conclusie en aanbevelingen

De vraag "*Hoe kunnen aan de hand van (de inhoud van) logbestanden, systeem- , beschikbaarheids- en reactie informatie problemen voorspeld worden?*" stond in dit onderzoek centraal.

Voor het verkrijgen en centraliseren van gegevens zijn er verschillende softwarepakketten beschikbaar. Door gegevens te centraliseren kunnen gegevens centraal (en eventueel gecombineerd) geanalyseerd worden. Fluentd en Logstash worden hiervoor geadviseerd te gebruiken. Deze softwarepakketten functioneren ongeveer het zelfde en kunnen beiden (de inhoud van) logbestanden en systeeminformatie verzamelen en centraliseren.

Gedurende het onderzoek bleek het echter dat aan de hand van (de inhoud van) logbestanden het moeilijk is problemen te voorspellen. Daarnaast was het al bekend bij het beantwoorden van de deelvraag omtrent het verkrijgen van gegevens, dat de applicaties die systeem informatie aanbieden hun beperkingen hebben. Deze applicaties sluiten goed aan op Fluentd en Logstash, maar het interval waarmee systeeminformatie wordt aangeboden is echter te groot om realtime voorspellingen te kunnen maken. Met behulp van de klassen die het .NET framework beschikbaar stelt kan een applicatie ontwikkeld worden die met een kleiner interval en eventueel meer of specifiekere informatie kan aanbieden.

Wanneer een applicatie ontwikkeld wordt die systeeminformatie aanbied, is het aanbieden van logbestanden via dezelfde applicatie ook interessant. Aan de hand van informatie uit de profielen van omgevingen is er bekend welke applicaties er op een omgeving uitgevoerd worden. Hierdoor wordt het bijvoorbeeld mogelijk te controleren of alle applicaties nog uitgevoerd worden op de betreffende omgeving. Daarnaast kan er zonder handmatige configuratie bepaald worden welke logbestanden er gemonitord moeten worden. Het is immers per applicatie bekend welke logbestanden er zijn en op welke locaties deze logbestanden standaard bewaard worden. De belasting van zowel de te monitoren omgeving en het centrale systeem kan door middel van een op maat ontwikkelde applicatie verkleint worden. De wijzingen in (de inhoud van) logbestanden kunnen samen met systeeminformatie verstuurd worden.

Bij het beantwoorden van hoe de deelvraag omtrent het herkennen van afwijkend gedrag werd duidelijk dat door patronen te ontdekken in gegevens het mogelijk wordt de toekomst te voorspellen. Met machine learning is het mogelijk te leren van gegevens. Dit resulteert in kennis die toegepast kan worden om verwacht gedrag te onderscheiden van afwijkend gedrag.

Tijdens het onderzoeken van de laatste deelvraag, omtrent voorspellen, werd het duidelijk dat veel problemen gekoppeld kunnen worden aan verwacht afwijkend gedrag. Dat betekend dat door afwijkend gedrag te herkennen, problemen herkent en mogelijk voorspeld kunnen worden. Door middel van Time Series Analysis kan het ontstaan van afwijkend gedrag voor gegevens uit systeem-, beschikbaarheids- en reactie informatie voorspeld worden. Door de verwachting te vergelijken met het daadwerkelijke gedrag wordt afwijkend gedrag bekend voordat problemen ontstaan. Hierdoor wordt het mogelijk problemen gekoppeld aan afwijkend gedrag uit dit soort gegevens te voorspellen voordat ze plaats vinden.

Er zijn ook problemen die gekoppeld zijn aan onverwacht afwijkend gedrag. Ondanks dat er een relatie is tussen het probleem en het onverwacht afwijkend gedrag is het echter niet mogelijk een voorspelling te maken. Op het moment dat het onverwacht afwijkend gedrag heeft plaats gevonden, heeft het probleem ook al plaats gevonden.

Uit (de inhoud van) logbestanden is het ook mogelijk afwijkend gedrag te herkennen maar dit is vaak te laat om nog voorspellingen te kunnen maken. Dit betekend niet dat (de inhoud van) logbestanden niet gebruikt kunnen worden om voorspellen mee te maken. Door statistieken toe te passen op de Severity van logberichten kan de verwachte frequentie waarop bepaalde logberichten ontstaan vergeleken worden met de daadwerkelijke frequentie.

9.2 Bijlage B - Onderzoeksrapport

Ondanks dat (de inhoud van) logbestanden moeilijker te analyseren zijn, is het centraal inzichtelijk maken van logberichten in een dashboard van toegevoegde waarde. Medewerkers hebben dan op een centrale plek inzicht op de staat van een omgeving.

6. Definities

Logbestand
Een tekstbestand waarin staat beschreven welke evenementen er hebben plaatsgevonden op een systeem *(Christensson, 2010)*.

Logbericht
De beschrijving van een evenement.

Systeeminformatie
Allesomvattend woord voor informatie over de hardware (zoals de temperatuur of het gebruik) en het softwareomgeving (zoals het besturingssysteem of de naam van het systeem) van een (machine op een) omgeving.

Afwijkend gedrag
Gedrag dat afwijkt van het gewenste gedrag.

Algoritme
Reeks instructies om vanaf een beginpunt een bepaald doel te bereiken *(Kernerman Dictionaries, z.d.)*.

Machine learning
Een studiegebied dat computers de mogelijkheid biedt te leren van gegevens zonder dat expliciet te programmeren *(Munoz, z.d.)*.

Probleem
Wanneer een systeem of applicatie anders functioneert dan gewenst.

Voorspelling
Het doen van uitspraken omtrent toekomstige gebeurtenissen en ontwikkelingen binnen een systeem, op grond van historisch gedrag van dat systeem *(Woorden-boek.nl, z.d.)*.

7. Literatuurlijst

Akhter Hossain, S., & Moniruzzaman, A. (2013). *NoSQL Database: New Era of Databases for Big data Analytics.* Opgeroepen op september 25, 2015, van http://arxiv.org/ftp/arxiv/papers/1307/1307.0191.pdf

Bare, C. (2012, juli 26). *Linear regression by gradient descent.* Opgeroepen op oktober 2, 2015, van http://digitheadslabnotebook.blogspot.nl/2012/07/linear-regression-by-gradient-descent.html

Busines Sstudies Online. (z.d.). *Forecasting.* Opgeroepen op oktober 2, 2015, van Busines Sstudies Online: http://www.businessstudiesonline.co.uk/AsA2BusinessStudies/TheoryNotes/2880/3Hr/PDF/04%20Time%20Series%20Analysis.pdf

C. Waclawski, A. (z.d.). *Autoregressive Integrated Moving Average Models for Comparing Forecasted to Actual.* Opgeroepen op oktober 2, 2015, van http://www2.sas.com/proceedings/sugi24/Stats/p286-24.pdf

Chappell, D. (2014). *Understanding NoSQL on Microsoft Azure.* Opgeroepen op september 25, 2015, van http://go.microsoft.com/fwlink/p/?linkid=330292&clcid=0x413

Christensson, P. (2010, april 14). *Log File Definition.* Opgeroepen op september 18, 2015, van Techterms: http://techterms.com/definition/logfile

Collectd. (z.d.). *Collectd – The system statistics collection daemon.* Opgeroepen op september 18, 2015, van Collectd.

Datastax. (2015, januari 2). *A Brief Introduction to Apache Cassandra.* Opgeroepen op september 25, 2015, van Datastax: https://academy.datastax.com/demos/brief-introduction-apache-cassandra

Datastax. (z.d.). *Configuring data consistency.* Opgeroepen op september 25, 2015, van Datastax: http://docs.datastax.com/en/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html

DB-Engines. (2015, september). *DB-Engines Ranking - popularity ranking of database management systems.* Opgeroepen op september 25, 2015, van DB-Engines: http://db-engines.com/en/ranking

DB-Engines. (z.d.). *InfluxDB System Properties.* Opgeroepen op september 25, 2015, van DB-Engines: http://db-engines.com/en/system/InfluxDB

DB-Engines. (z.d.). *Time Series DBMS.* Opgeroepen op september 25, 2015, van DB-Engines: http://db-engines.com/en/article/Time+Series+DBMS

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., . . . Vogels, W. (2007). *Dynamo: Amazon's Highly Available Key-value Store.* Opgeroepen op september 25, 2015, van http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf

Di Bucchianico, A. (z.d.). *Technische Universiteit Eindhoven.* Opgeroepen op oktober 6, 2015, van Enkelvoudige lineaire regressie: http://www.win.tue.nl/~adibucch/2DS00/Enkelvoudige%20lineaire%20regressie.pdf

Elastic. (z.d.). *Elasticsearch: RESTful, Distributed Search & Analytics.* Opgeroepen op september 25, 2015, van Elastic: https://www.elastic.co/products/elasticsearch

Elastic. (z.d.). *File.* Opgeroepen op september 18, 2015, van Elastic: https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html

Elastic. (z.d.). *Logstash: Collect, Parse, Transform Logs.* Opgeroepen op september 18, 2015, van Elastic: https://www.elastic.co/products/logstash

Elastic. (z.d.). *Output plugins.* Opgeroepen op september 25, 2015, van Elastic: https://www.elastic.co/guide/en/logstash/current/output-plugins.html

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). *From Data Mining to Knowledge Discovery in Databases.* Opgeroepen op oktober 2, 2015, van http://www.csd.uwo.ca/faculty/ling/cs435/fayyad.pdf

Fluentd. (2015, april 2). *Tail Input Plugin.* Opgeroepen op september 18, 2015, van Fluentd: http://docs.fluentd.org/articles/in_tail

Fluentd. (z.d.). *List of Data Outputs.* Opgeroepen op september 25, 2015, van Fluentd: http://www.fluentd.org/dataoutputs

Fluentd. (z.d.). *Open source data collector.* Opgeroepen op oktober 2, 2015, van Fluentd: http://www.fluentd.org/

Fluentd. (z.d.). *Open Source Data Collector | Unified Logging Layer.* Opgeroepen op september 18, 2015, van Fluentd: http://www.fluentd.org/

9.2 Bijlage B - Onderzoeksrapport

Fuqua School of Business. (z.d.). *Forecasting with seasonal adjustment and linear exponential.* Opgeroepen op oktober 2, 2015, van Fuqua School of Business: https://faculty.fuqua.duke.edu/~rnau/Decision411_2007/411outbd.htm

Gerhards, R. (2009, maart). *The Syslog Protocol.* Opgeroepen op september 18, 2015, van https://tools.ietf.org/html/rfc5424

Gilbert, S., & Lynch, N. (2002, juni). *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services.* Opgeroepen op september 25, 2015, van https://www.comp.nus.edu.sg/~gilbert/pubs/BrewersConjecture-SigAct.pdf

Graylog. (z.d.). *Open source log management.* Opgeroepen op oktober 2, 2015, van Graylog: https://www.graylog.org/

Graylog. (z.d.). *Open Source Log Management with Graylog.* Opgeroepen op september 18, 2015, van Graylog: https://www.graylog.org/

Humanoriented. (z.d.). *Mining for Groups Using Clustering Algorithms.* Opgeroepen op oktober 2, 2015, van Humanoriented: http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/mvoget/cluster/cluster.htm

Hurst, N. (2010, maart 15). *Visual Guide to NoSQL Systems.* Opgeroepen op september 25, 2015, van Nathan Hurst's Blog: http://blog.nahurst.com/visual-guide-to-nosql-systems

InfluxDB. (z.d.). *Data Exploration.* Opgeroepen op september 25, 2015, van InfluxDB: https://influxdb.com/docs/v0.9/query_language/data_exploration.html

InfluxDB. (z.d.). *Storage Engine.* Opgeroepen op september 25, 2015, van https://influxdb.com/docs/v0.9/concepts/storage_engine.html

InfluxDB. (z.d.). *What's in a database?* Opgeroepen op september 25, 2015, van InfluxDB: https://influxdb.com/docs/v0.9/concepts/crosswalk.html

Info Support. (z.d.). *Over Info Support.* Opgeroepen op september 14, 2015, van http://www.infosupport.com/info-support-b-v/

Kernerman Dictionaries. (z.d.). *Algoritme Nederlands woordenboek.* Opgeroepen op oktober 2, 2015, van Kernerman Dictionaries: http://www.woorden.org/woord/algoritme

Khan Zaki, A. (2014, mei). *NoSQL databases: New millennium database for big data.* Opgeroepen op september 25, 2015, van http://esatjournals.org/Volumes/IJRET/2014V03/I15/IJRET20140315080.pdf

Lipcon, T. (2009, juni 11). *Design Patterns for Distributed Non-Relational Databases.* Opgeroepen op september 25, 2015, van Slideshare: http://www.slideshare.net/guestdfd1ec/design-patterns-for-distributed-nonrelational-databases

Lison, P. (2012, oktober 3). *An introduction to machine learning.* Opgeroepen op oktober 2, 2015, van http://folk.uio.no/plison/pdfs/talks/machinelearning.pdf

Loganalyzer. (z.d.). *Apache/NCSA Common Log Format.* Opgeroepen op september 18, 2015, van Loganalyzer: http://www.loganalyzer.net/log-analyzer/apache-common-log.html

Logentries logo. (z.d.). Opgeroepen op oktober 2, 2015, van DZone: https://dz2cdn1.dzone.com/storage/partner-logo/41764-logentries-logo.png

Logentries. (z.d.). *Log Management & Analysis Software Made Easy.* Opgeroepen op september 18, 2015, van Logentries: https://logentries.com/

Logentries. (z.d.). *Syslog Logging.* Opgeroepen op september 18, 2015, van Logentries: https://logentries.com/doc/about-syslog/

Logentries. (z.d.). *Tags & Alerts.* Opgeroepen op september 18, 2015, van Logentries: https://logentries.com/doc/tagsalerts/

Logstash logo. (z.d.). Opgeroepen op oktober 2, 2015, van Pagerduty: https://www.pagerduty.com/assets/logstash_logo.png

Makanju, A., Nur Zincir-Heywood, A., & Evangelos, E. (2009, juni 26). *A Lightweight Algorithm for Message Type Extraction in Event Logs.* Opgeroepen op oktober 2, 2015, van https://www.cs.dal.ca/sites/default/files/technical_reports/CS-2009-07.pdf

McCandless, M., Hatcher, E., & Gospodnetić, O. (2010). *Lucene in Action, Second Edition.* Manning Publications. Opgehaald van http://dl.ftl.vn/share/book/Manning%20Lucene%20in%20Action%202nd%20Edition.pdf

9.2 Bijlage B - Onderzoeksrapport

Microsoft Developer Network. (z.d.). *DriveInfo Class.* Opgeroepen op september 18, 2015, van Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/system.io.driveinfo(v=vs.110).aspx

Microsoft Developer Network. (z.d.). *Environment Class.* Opgeroepen op september 18, 2015, van Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/system.environment(v=vs.110).aspx

Microsoft Developer Network. (z.d.). *Win32 Classes.* Opgeroepen op september 18, 2015, van Microsoft Developer Network: https://msdn.microsoft.com/en-us/library/aa394084(v=vs.85).aspx

Microsoft Technet. (z.d.). *W3C Extended Log File Format (IIS 6.0).* Opgeroepen op september 18, 2015, van Microsoft Technet: http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/676400bc-8969-4aa7-851a-9319490a9bbb.mspx?mfr=true

Middelkoop, R. (2015, februari 4). *Workshop2 Plannen Verzamelen.* Opgeroepen op september 14, 2015, van https://github.com/HANICA/mad-research/blob/master/les2-plannen-verzamelen/Workshop2-Plannen%2BVerzamelen.pptx

MongoDB. (z.d.). *Do What You Could Never Do Before.* Opgeroepen op september 25, 2015, van MongoDB: https://www.mongodb.com/what-is-mongodb

MongoDB. (z.d.). *Real-Time Analytics.* Opgeroepen op september 25, 2015, van MongoDB: https://www.mongodb.com/use-cases/real-time-analytics

Munoz, A. (z.d.). *Machine Learning and Optimization.* Opgeroepen op oktober 2, 2015, van https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf

Naseri, M. (2013, januari 31). *System Information.* Opgeroepen op september 18, 2015, van Code Project: http://www.codeproject.com/Articles/362227/System-Information

Oates, B. J. (2006). *Researching Information Systems and Computing.* London: SAGE Publications Ltd. Opgeroepen op september 14, 2015

OpenTSDB. (z.d.). *How does OpenTSDB work.* Opgeroepen op september 25, 2015, van OpenTSDB: http://opentsdb.net/overview.html

Redis. (z.d.). *An introduction to Redis data types and abstractions.* Opgeroepen op september 25, 2015, van Redis: http://redis.io/topics/data-types-intro

Redis. (z.d.). *Expire key seconds.* Opgeroepen op september 25, 2015, van Redis: http://redis.io/commands/expire

Redis. (z.d.). *Introduction to Redis.* Opgeroepen op september 25, 2015, van Redis: http://redis.io/topics/introduction

Redis. (z.d.). *Pub/Sub.* Opgeroepen op september 25, 2015, van Redis: http://redis.io/topics/pubsub

Schuster, A. (2007, juni 15). *Introducing the Microsoft Vista event log file format.* Opgeroepen op september 18, 2015, van http://www.dfrws.org/2007/proceedings/p65-schuster.pdf

Splunk logo. (z.d.). Opgeroepen op oktober 2, 2015, van Lucidworks: http://lucidworks.com/wp-content/themes/lucidworks/library/img/logos/integrations/splunk.png

Splunk. (z.d.). *Operational Intelligence, Log Management, Application Management, Enterprise Security and Compliance.* Opgeroepen op september 18, 2015, van Splunk: http://www.splunk.com/

SSC Server. (z.d.). *Features.* Opgeroepen op september 18, 2015, van SSC Server: http://ssc-serv.com/

Turnhout, K., Craenmehr, S., Holwerda, R., Menijn, M., Zwart, J.-P., & Bakker, R. (2013). *Triangulatie, een basis voor de onderzoeksleerlijn in ict en media onderwijs.* Opgeroepen op september 14, 2015, van NIOC - ontmoetingsplaats voor informatica-onderwijs: http://nioc.nl/archief/2013/wp-content/uploads/2013/12/nioc-paper-definitief-KvT.pdf

Wageningen Universiteit. (2013, februari 19). *Lineaire regressie.* Opgeroepen op oktober 2, 2015, van Wageningen Universiteit: https://www.wageningenur.nl/upload_mm/5/8/4/63876140-57e9-466b-bc5f-cd71b91bb72e_07_regressie_brf(19-02-2013)ll.pdf

Woorden-boek.nl. (z.d.). *Voorspelling.* Opgeroepen op oktober 6, 2015, van Woorden-boek.nl: http://www.woorden-boek.nl/woord/voorspelling

ŷhat. (2013, juli 25). *Intuitive Classification using KNN and Python.* Opgeroepen op oktober 2, 2015, van ŷhat: http://blog.yhathq.com/posts/classification-using-knn-and-python.html

Zhang, Z. (2012, maart 22). *K-means Algorithm - Cluster Analysis in Data Mining.* Opgeroepen op oktober 2, 2015, van http://user.engineering.uiowa.edu/~ie_155/lecture/K-means.pdf

Functioneel Ontwerp

Predicting Server Flaws

Anthony Huebers

Functioneel Ontwerp

Predicting Server Flaws

Titel	Functioneel Ontwerp
Project/Onderwerp	Predicting Server Flaws
Versie	1.3
Status	Definitief
Datum	26-nov-2015
Bestand	Functioneel Ontwerp - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

9.3 Bijlage C - Functioneel Ontwerp

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	12-10-2015	Anthony Huebers	Creatie
1.1	Concept	29-10-2015	Anthony Huebers	Sprint 1
1.2	Concept	02-11-2015	Anthony Huebers	Sprint 2
1.3	Definitief	26-11-2015	Anthony Huebers	Sprint 3 + Korte sprint

Distributie

Versie	Status	Datum	Aan

Referenties

Code	Bron

9.3 Bijlage C - Functioneel Ontwerp

Inhoudsopgave

9.3 Bijlage C - Functioneel Ontwerp

1. Inleiding

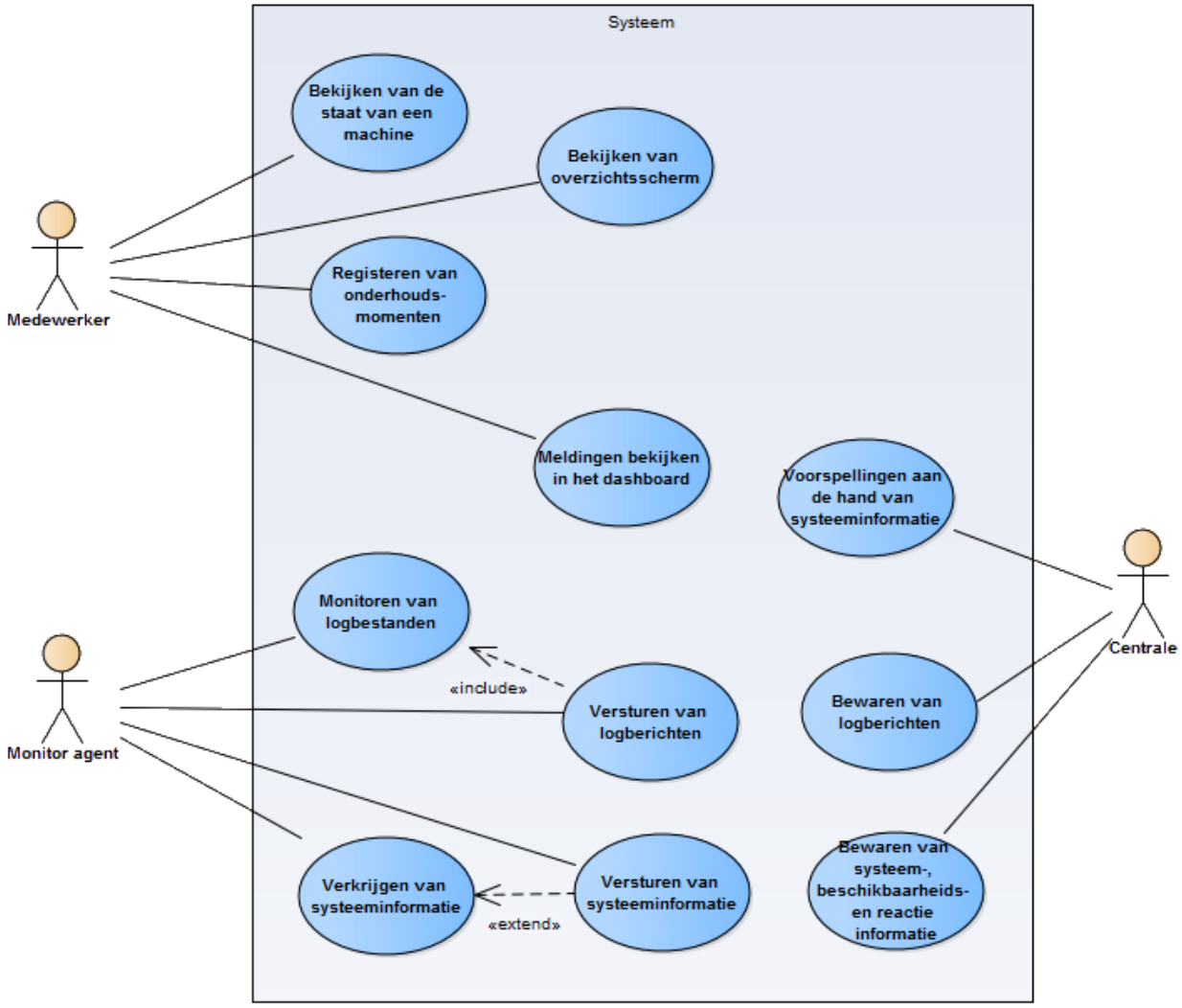
Binnen het project wordt een (proof of concept) systeem gerealiseerd wat (patronen in) problemen kan herkennen en mogelijk problemen in de toekomst kan voorspellen. Dit document bevat de functionele specificaties van het (proof of concept) systeem.

Het doel van het (proof of concept) systeem is het bewijzen van de onderzochte mogelijkheden.

1.1 Afhankelijkheden

Er zijn op dit moment geen externe afhankelijkheden.

2. Use Cases Diagram



Figuur 1 - Use Case Diagram

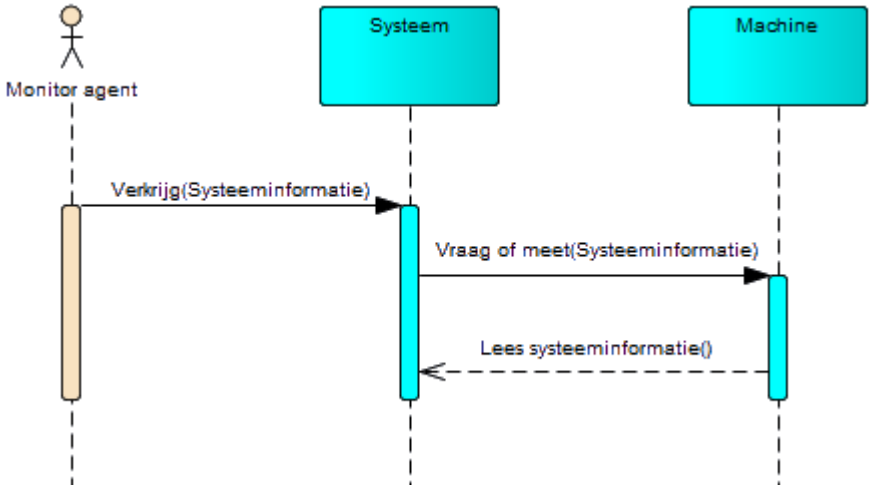
3. User Stories

De beschreven user stories in dit hoofdstuk staat op volgorde van realisatie.

3.1 Verkrijgen van systeeminformatie

Titel	Verkrijgen van systeeminformatie
ID	UC01
User story	Als monitor agent, wil ik de systeeminformatie van een machine weten zodat ik weet hoe een machine presteert.
Beschrijving	1. De monitor agent geeft aan welke systeeminformatie verkregen moet worden. 2. Het systeem vraagt de huidige waardes van de specificeerde systeeminformatie op. 3. Het systeem leest de opgevraagde systeeminformatie.
Uitzondering	3A. Een of meerdere specificeerde systeeminformatie kan niet worden opgevraagd.
Resultaat	De huidige waardes van de specificeerde systeeminformatie van een machine is bekend.

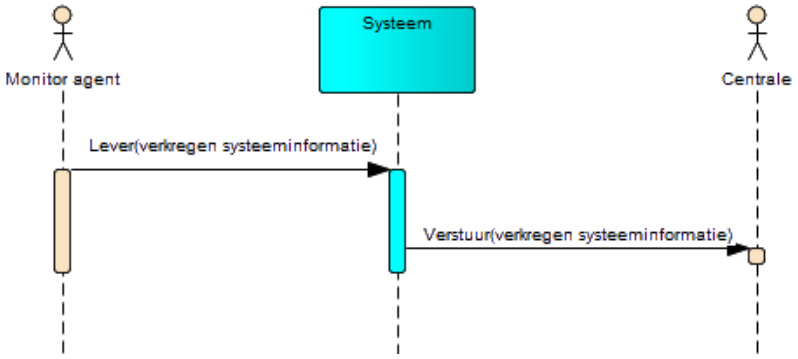
Acceptatie criteria	1. De meest recente systeeminformatie is zichtbaar en bevat:
	- Processor verbruik als percentage
	- Geheugen verbruik als percentage
	- Netwerk verbruik in kilo bytes per seconde
	  - Ontvangen gegevens
	  - Verstuurden gegevens
	- Harde schijf verbruik (per fysieke schijf)
	  - Lees operaties in mega bytes per seconde
	  - Schrijf operaties in mega bytes per seconde
	  - Beschikbare ruimte (per partitie) als percentage



Figuur 2 - Sequence Diagram "UC01 - Verkrijgen van systeeminformatie"

3.2 Versturen van systeeminformatie

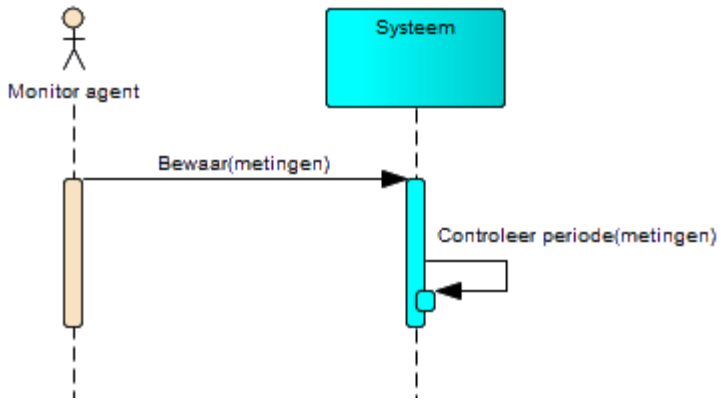
Titel	Versturen van systeeminformatie
ID	UC02
User story	Als monitor agent, wil ik graag de verkregen systeeminformatie van een machine versturen naar de centrale zodat de centrale de verkregen systeeminformatie kan bewaren.
Beschrijving	1. De monitor agent levert de verkregen systeeminformatie. 2. Het systeem verstuurt de geleverde systeeminformatie naar de centrale.
Uitzondering	2A. De centrale is ontoegankelijk
Resultaat	De geleverde systeeminformatie zijn verstuurd naar de centrale.
Acceptatie criteria	1. De monitor agent communiceert met de centrale door middel van JSON over HTTP(S).



Figuur 3 - Sequence Diagram "UC02 - Versturen van systeeminformatie"

3.3 Bewaren van systeem-, beschikbaarheids- en reactie informatie

Titel	Bewaren van systeem-, beschikbaarheids- en reactie informatie
ID	UC03
User story	Als monitor agent, wil ik graag de verkregen systeem-, beschikbaarheids- en reactie informatie bewaren zodat ik deze kan verzamelen en hier analyse op kan doen.
Beschrijving	1. De monitor agent levert de verkregen systeem-, beschikbaarheids- en reactie informatie (metingen) van een machine. 2. Het systeem zoekt voor de gegeven machine de datum waarop de metingen voor het laatst zijn opgeslagen. 3. Het systeem bewaard de gegeven metingen.
Uitzondering	3A. De periode van de verkregen metingen valt voor de laatst bekende datum.
Resultaat	Metingen zijn bewaard.
Acceptatie criteria	1. Een meting uit systeem-, beschikbaarheids- en reactie informatie wordt opgeslagen in InfluxDB en bevat: - Tag genaamd "host" met de naam van de machine - Waarde van een meting - Datum en tijd van meting (UTC) 2. InfluxDB moet indien nodig vervangen kunnen worden door een andere database implementatie.

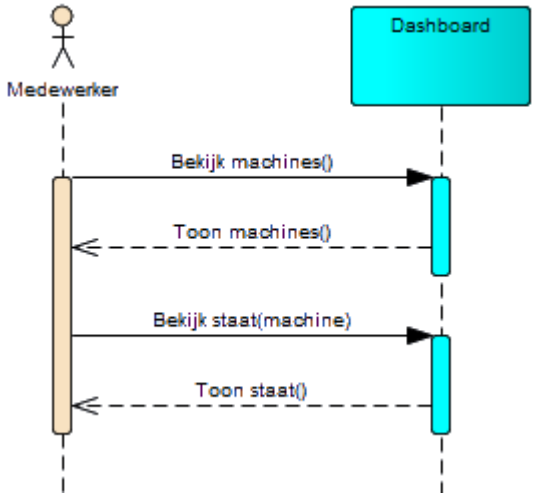


Figuur 4 - Sequence Diagram "UC03 - Bewaren van systeem-, beschikbaarheids- en reactie informatie"

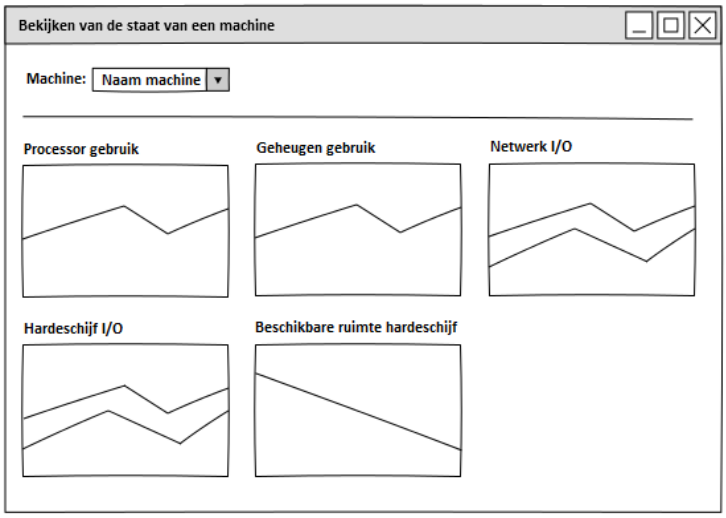
3.4 Bekijken van de staat van een machine

Titel	Bekijken van de staat van een machine
ID	UC04
User story	Als medewerker, wil ik de staat van een machine kunnen bekijken zodat ik kan controleren of een machine nog het gewenste gedrag vertoont.
Beschrijving	1. De medewerker selecteert een machine. 2. Het systeem verzameld de metingen uit systeeminformatie van de geselecteerde machine voor het afgelopen uur. 3. Het systeem toont de verzamelde metingen.
Uitzondering	N.v.t.
Resultaat	Alle metingen van de geselecteerde machine voor het afgelopen uur zijn getoond.

Acceptatie criteria	1. De periode (afgelopen uur) en startmoment (gister tot nu) zijn instelbaar.
	2. De metingen van een machine worden gevisualiseerd in een lijn grafiek.
	3. Staat van een machine bevat:
	- Processor verbruik als percentage
	- Geheugen verbruik als percentage
	- Netwerk verbruik in kilo bytes per seconde
	  - Ontvangen gegevens
	  - Verstuurden gegevens
	- Harde schijf verbruik (per fysieke schijf)
	  - Lees operaties in mega bytes per seconde
	  - Schrijf operaties in mega bytes per seconde
	  - Beschikbare ruimte (per partitie) als percentage



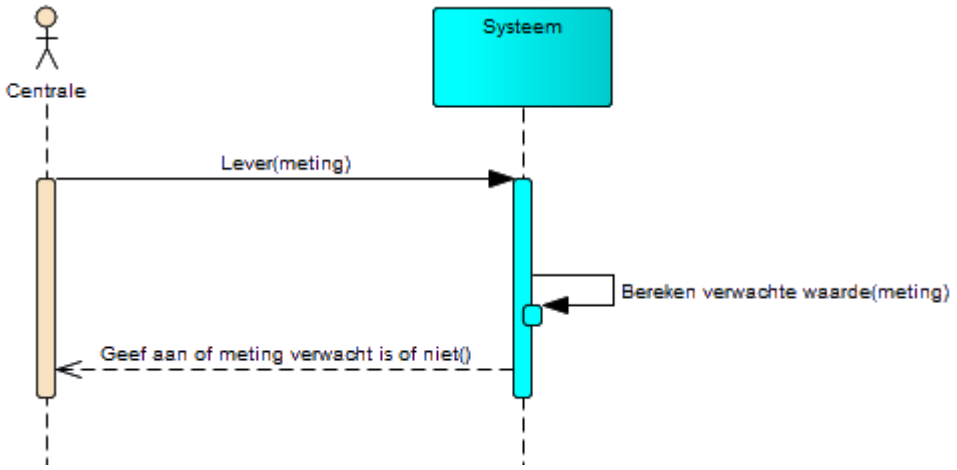
Figuur 5 - Sequence Diagram "UC04- Bekijken van de staat van een machine"



Figuur 6 - Schermschets "UC04 - Bekijken van de staat van een machine"

3.5 Voorspellingen aan de hand van systeeminformatie

Titel	Voorspellingen aan de hand van systeeminformatie
ID	UC05
User story	Als centrale wil ik graag op basis van systeeminformatie voorspellen maken zodat ik kan zien of een machine onverwacht gedrag vertoont.
Beschrijving	1. De voorspeller levert een meting van een machine. 2. Het systeem berekend de verwachte waarde van een meting. 3. Het systeem geeft aan of de gegeven meting verwacht wordt.
Uitzondering	N.v.t.
Resultaat	Voor gegeven metingen is het bekend of deze verwacht wordt. Wanneer deze niet verwacht wordt kan geconcludeerd worden dat een machine onverwacht gedrag vertoont.
Acceptatie criteria	1. Voorspellingen voor processor en geheugen verbruik - Niet langer dan 30 seconden boven of gelijk aan 80% - Gebruik mag niet langer dan 30 seconden meer dan 50% na onder of na boven afwijken ten opzichte van het gemiddelde verbruik voor de vorige periode van een uur. 2. Alle parameters (tijd en percentages) moeten aanpasbaar zijn.

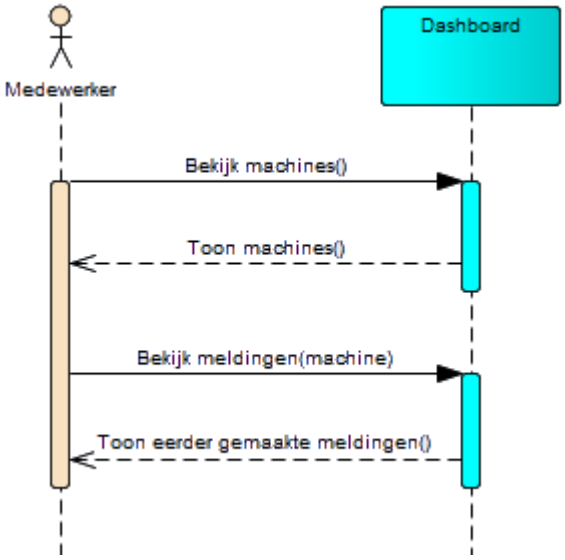


Figuur 7 - Sequence Diagram "UC05 - Voorspellingen aan de hand van systeeminformatie"

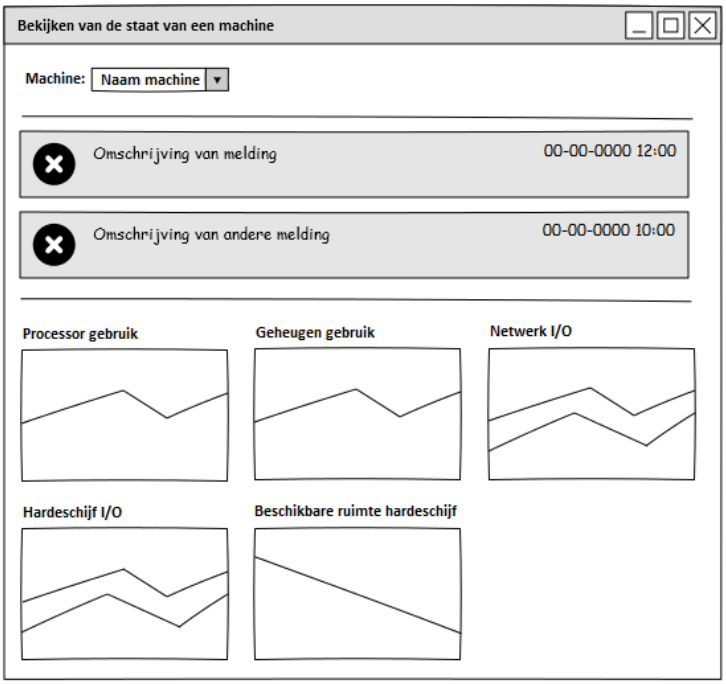
3.6 Meldingen bekijken in het dashboard

Titel	Meldingen bekijken in het dashboard
ID	UC06
User story	Als medewerker wil ik graag de gemaakte meldingen bekijken in het dashboard zodat gemaakte voorspellingen zichtbaar zijn.
Beschrijving	1. De medewerker selecteert een machine. 2. Het systeem verzamelt eerder gemaakte meldingen voor de geselecteerde machine voor de afgelopen week. 3. Het systeem toont de gemaakte meldingen. 4. Het systeem wacht tot een melding gemaakt wordt. 5. Het systeem toont de gemaakte melding
Uitzondering	N.v.t.
Resultaat	Gemaakte meldingen zijn zichtbaar en zijn realtime inzichtelijk.
Acceptatie criteria	1. Eerder gemaakte meldingen (met een geschiedenis van 1 week) zijn inzichtelijk. - De geschiedenis periode is instelbaar - Eerder gemaakte meldingen zijn gesorteerd van meest naar minst recente melding. 2. Meldingen gemaakt door de voorspeller worden gepusht naar het dashboard. 3. Melding bevat de volgende gegevens: 2A. De naam van de machine (hostname) waarover de melding is gemaakt. 3A. De datum en tijd waarop de melding is gemaakt. 4A. Omschrijving waarom de melding is gemaakt door de voorspeller.

9.3 Bijlage C - Functioneel Ontwerp



Figuur 8 - Sequence Diagram "UC06 - Meldingen bekijken in het dashboard"



Figuur 9 - Schermschets "UC06 - Meldingen bekijken in het dashboard"

3.7 Uitbreiding verkrijgen van systeeminformatie

Titel	Uitbreiding verkrijgen van systeeminformatie
ID	UC01-1
User story	Als monitor agent, wil ik de systeeminformatie van een machine uitbreiden met het verbruik van uitgevoerde processen zodat ik nog beter weet hoe een machine presteert.
Beschrijving	1. De monitor agent geeft aan welke (actieve) processen er zijn. 2. Het systeem vraagt de huidige systeeminformatie waardes op voor de specificeerde processen. 3. Het systeem leest de opgevraagde systeeminformatie.
Uitzondering	2A. Een proces wordt uitgevoerd door het systeem.
Resultaat	De huidige systeeminformatie waardes voor de specificeerde processen uitgevoerd op een machine zijn bekend.
Acceptatie criteria	1. Alle processen die niet uitgevoerd worden door het systeem. 2. Extra systeeminformatie bevat: - Processor verbruik als percentage per proces. - Geheugen verbruik als percentage per proces.

3.8 Test data generatie

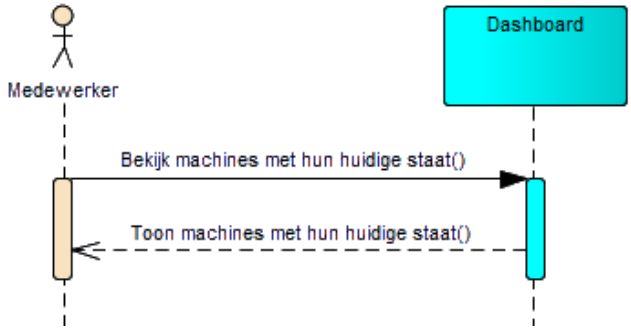
Titel	Test data generatie
ID	UC01-2
User story	Als ontwikkelaar heb ik test data nodig om een betere gedragsherkenningservice te kunnen realiseren zodat voorspellingen accurater zijn.
Beschrijving	1. De ontwikkelaar start het systeem op een machine. 2. Het systeem vraagt de systeeminformatie van de machine op. 3. Het systeem schrijft de verkregen systeeminformatie weg naar een bestand
Uitzondering	N.v.t.
Resultaat	De verkregen systeeminformatie van een machine worden weggeschreven naar een bestand
Acceptatie criteria	1. Hiervoor wordt door de ontwikkelaar een console tool aangeleverd aan de opdrachtgever - Test metingen moeten lokaal naar een bestand weggeschreven worden. - Er moet per dag een nieuw bestand gemaakt worden. 2. Alle (door de monitor agent te lezen) systeeminformatie van een of meerdere machines voor minimaal 1 week.

3.9 Dashboard (detail) dynamische bijwerken

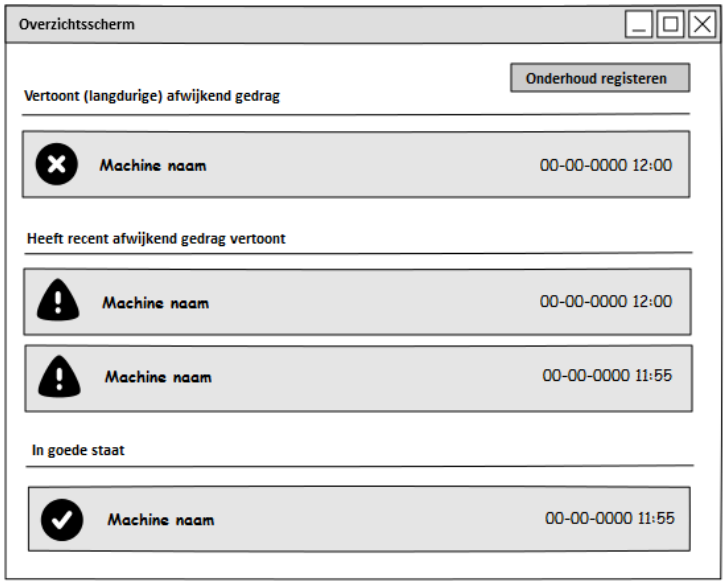
Titel	Dashboard (detail) dynamische bijwerken
ID	UC06-1
User story	Als medewerker wil ik graag dat het dashboard automatische bijgewerkt wordt zodat ik realtime de staat van een machine kan bekijken.
Beschrijving	1. De medewerker selecteert een machine. 2. Het systeem toont realtime de staat van de gespecifieerde machine.
Uitzondering	N.v.t.
Resultaat	Alle metingen van de geselecteerde machine voor het afgelopen uur zijn getoond en worden realtime bijgewerkt.
Acceptatie criteria	1. Grafieken moeten realtime bijgewerkt blijven.

3.10 Bekijken van overzichtsscherm

Titel	Bekijken van overzichtsscherm
ID	UC07
User story	Als medewerker wil ik graag een overzichtsscherm waarop ik een lijst van alle machines en hun huidige staat kan bekijken zodat ik in een oogopslag kan zien waar actie op ondernomen moet worden.
Beschrijving	1. De medewerker start het dashboard 2. Het systeem vraagt alle machines en hun huidige staat op. 3. Het systeem sorteert alle machines en hun huidige staat. 4. Het systeem toont alle machines en hun huidige staat.
Uitzondering	N.v.t.
Resultaat	Een lijst van alle machines en hun huidige staat is weergeven.
Acceptatie criteria	1. Kleurcodering van de staat van een machine - Groen wanneer er niks aan de hand is - Oranje wanneer een afwijking gedetecteerd is in de laatste 90 seconden - Rood wanneer er langer dan 10 seconden afwijkend gedrag gedetecteerd wordt. 2. Realtime bij werken van de status 3. Sortering van de staat van machines - Rood bovenaan, dan oranje en als laatste groen - Realtime bijwerken van deze volgorde waarbij alleen de verandering van status invloed heeft op de volgorde zodat de machines niet blijven verspringen maar dat een nieuwe rode onder een eerdere rode verschijnt.



Figuur 10 - Sequence Diagram "UC07 - Bekijken van overzichtsscherm"



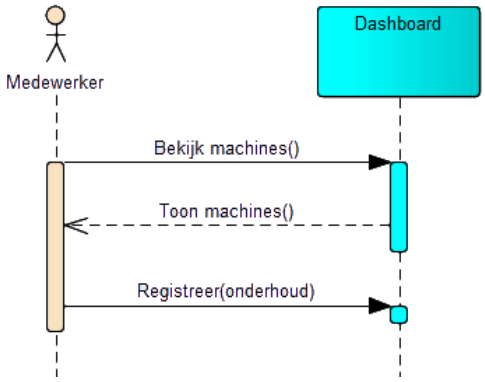
Figuur 11 - Schermschets "UC07 - Bekijken van overzichtsscherm"

3.11 Registeren van onderhoudsmomenten

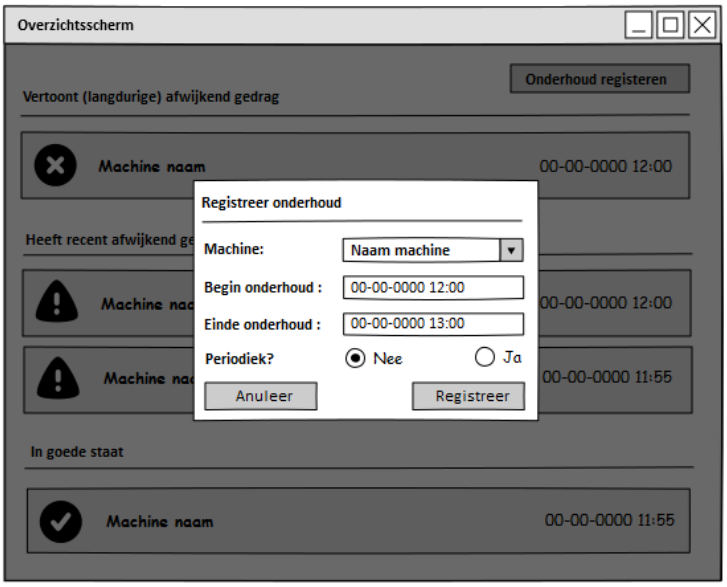
Titel	Registeren van onderhoudsmomenten
ID	UC08
User story	Als medewerker wil ik graag kunnen aangeven voor welke periode er onderhoud gepleegd wordt op een specifieke machine zodat de voorspeller dit niet ziet als afwijkend gedrag.
Beschrijving	1. De medewerker specificeert een machine en periode wanneer onderhoud gepleegd gaat worden. 2. Het systeem bewaard de gespecifieerde onderhoudsgegevens.
Uitzondering	N.v.t.
Resultaat	De onderhoudsgegevens zijn bewaard.

9.3 Bijlage C - Functioneel Ontwerp

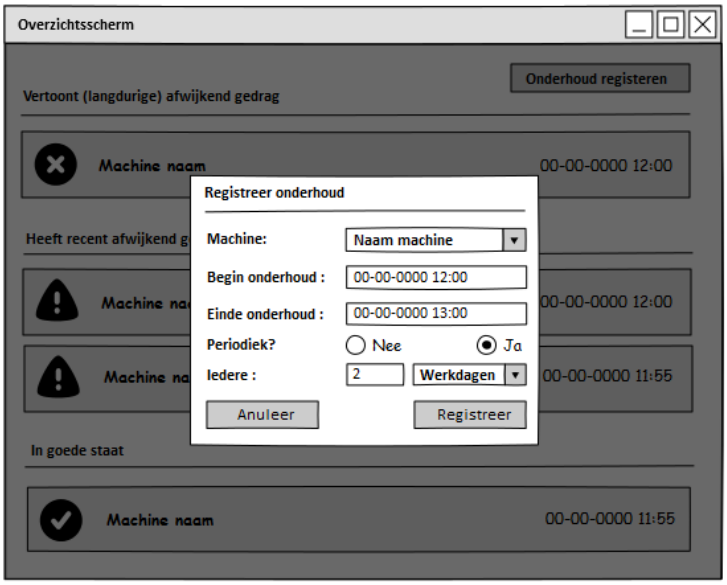
Titel	Registeren van onderhoudsmomenten
Acceptatie criteria	1. In het dashboard moet aangegeven kunnen worden: - Op welke machine onderhoud gepleegd gaat worden. - De datum en tijd wanneer het onderhoud gepleegd wordt. - De datum en tijd wanneer het onderhoud afgerond is. - Of het onderhoud herhaaldelijk is. 2. Wanneer een onderhoudsmoment herhaaldelijk is, moet aangegeven kunnen worden: - De cyclus van het onderhoud, iedere x aantal (dagen / werkdagen / weken / maanden)



Figuur 12 - Sequence Diagram "UC08 - Registeren van onderhoudsmomenten"



Figuur 13 – Schermschets 1 "UC08 - Registeren van onderhoudsmomenten"



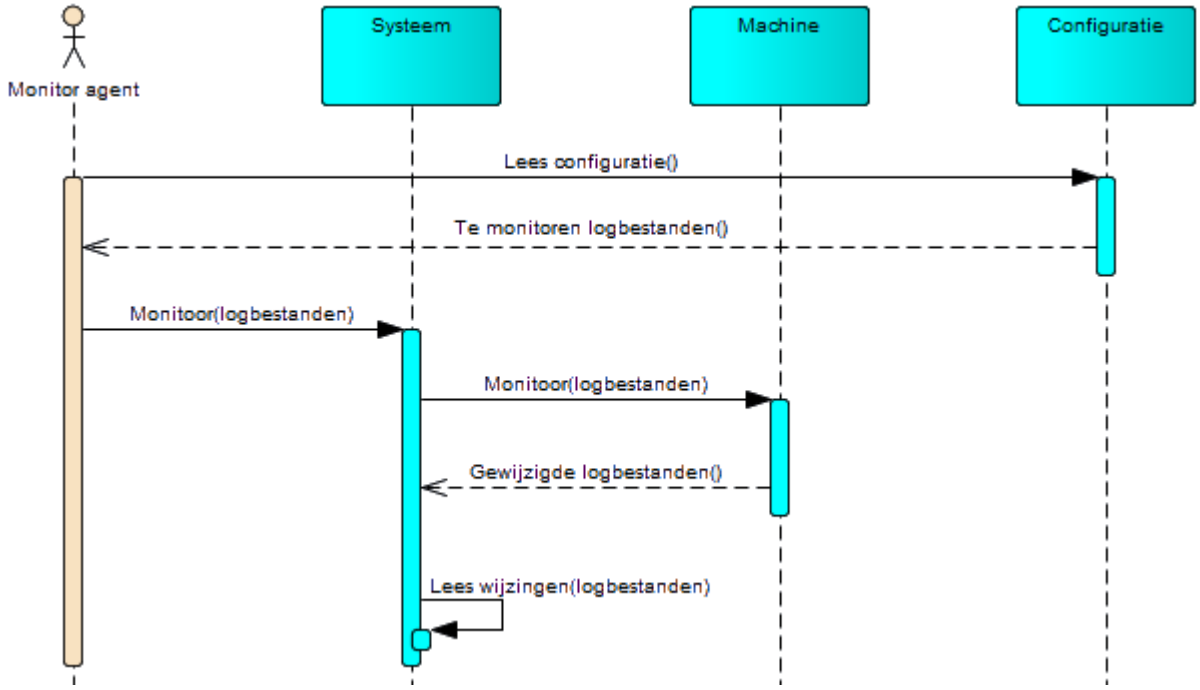
Figuur 14 - Schermschets 2 "UC08 - Registeren van onderhoudsmomenten"

3.12 Monitoren van logbestanden

Titel	Monitoren van logbestanden
ID	UC09
User story	Als monitor agent, wil ik graag (de inhoud van) logbestanden monitoren zodat ik meer inzicht krijg in de staat van applicaties op een machine.
Beschrijving	1. De monitor agent leest de configuratie 2. De monitor agent geeft aan welke logbestanden in de gaten gehouden moeten worden. 3. Het systeem controleert voor ieder logbestand of deze gewijzigd is. 4. Het systeem leest de wijzingen van logbestanden.
Uitzondering	4A. Er zijn geen logbestanden gewijzigd.
Resultaat	Wijzingen in logbestanden zijn gedetecteerd en ingelezen.

9.3 Bijlage C - Functioneel Ontwerp

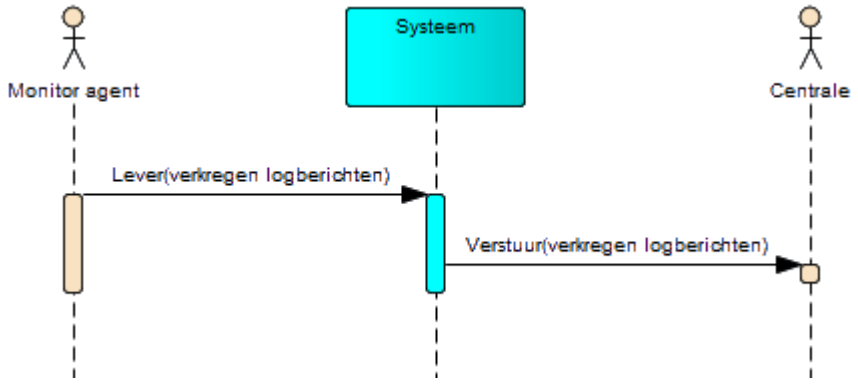
Titel	Monitoren van logbestanden
Acceptatie criteria	1. Logbestanden van de onderstaande applicaties moeten ingelezen kunnen worden: - Team Foundation Server (TFS) - Jenkins - JIRA - Stash - SonarQube - Nexus - Confluence - MSSQL - Internet Information Services (IIS) - NGINX - Sharepoint - Docker 2. Ieder logbericht bevat: - Naam van de applicatie die het logbericht gegeneerd heeft - Het severity level - Het bericht - Datum en tijd van een logbericht



Figuur 15 - Sequence Diagram "UC09 - Monitoren van logbestanden"

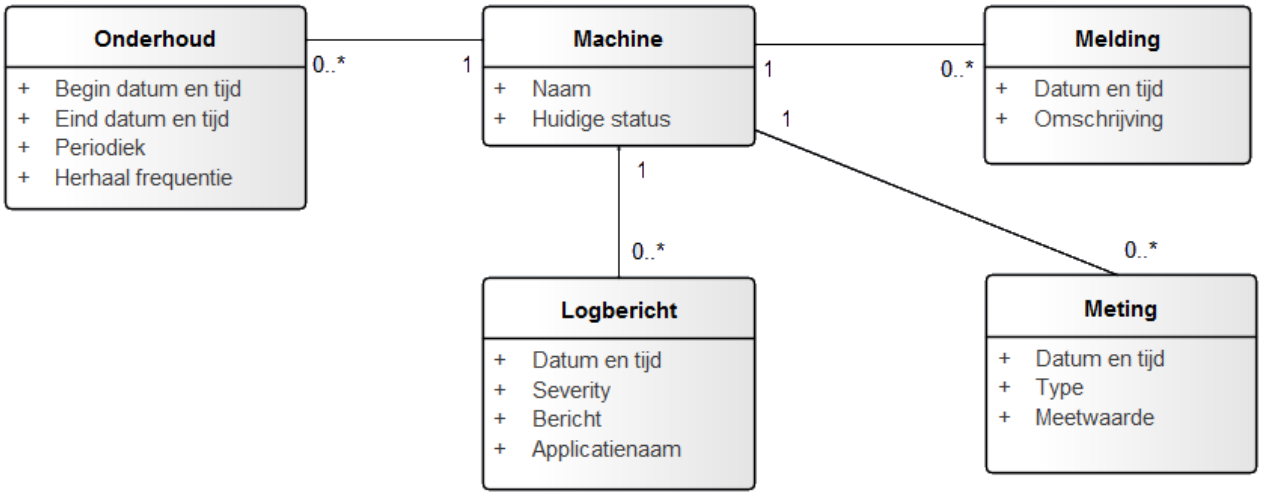
3.13 Versturen van logberichten

Titel	Versturen van logberichten
ID	UC10
User story	Als monitor agent, wil ik graag de verkregen logberichten van een machine versturen naar de centrale zodat de centrale de verkregen logberichten kan bewaren.
Beschrijving	1. De monitor agent levert de verkregen logberichten. 2. Het systeem verstuurt de geleverde logberichten naar de centrale.
Uitzondering	2A. De centrale is ontoegankelijk
Resultaat	De geleverde logberichten zijn verstuurd naar de centrale.
Acceptatie criteria	1. De monitor agent communiceert met de centrale door middel van JSON over HTTP(S).



Figuur 16 - Sequence Diagram "UC10 – Versturen van logberichten"

4. Domeinmodel



Figuur 17 - Conceptual Model

Technisch Ontwerp

Predicting Server Flaws

Anthony Huebers

Technisch Ontwerp

Predicting Server Flaws

Titel	Technisch Ontwerp
Project/Onderwerp	Predicting Server Flaws
Versie	1.3
Status	Definitief
Datum	27-nov-2015
Bestand	Technisch Ontwerp - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

9.4 Bijlage D - Technisch Ontwerp

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	14-10-2015	Anthony Huebers	Creatie
1.1	Concept	29-10-2015	Anthony Huebers	Sprint 1
1.2	Concept	03-11-2015	Anthony Huebers	Sprint 2
1.3	Concept	27-11-2015	Anthony Huebers	Sprint 3 + Korte sprint

Distributie

Versie	Status	Datum	Aan

Referenties

Code	Bron

9.4 Bijlage D - Technisch Ontwerp

Inhoudsopgave

9.4 Bijlage D - Technisch Ontwerp

1. Inleiding

Binnen het project wordt een (proof of concept) systeem gerealiseerd wat (patronen in) problemen kan herkennen en mogelijk problemen in de toekomst kan voorspellen. Dit document bevat de technische specificaties van het (proof of concept) systeem.

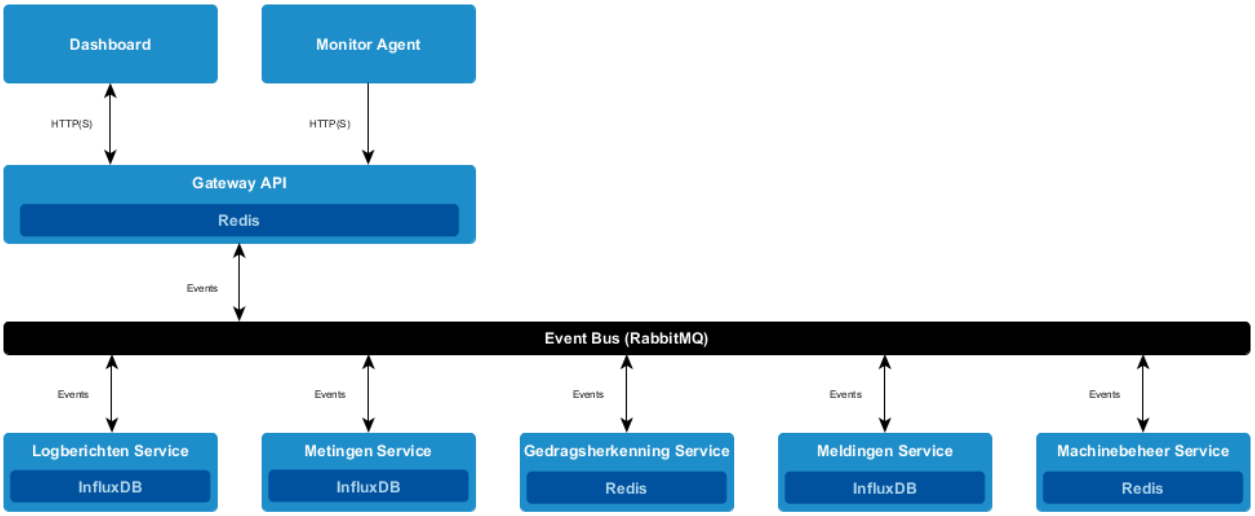
Het doel van het (proof of concept) systeem is het bewijzen van de onderzochte mogelijkheden.

1.1 Afhankelijkheden

Er zijn op dit moment geen afhankelijkheden.

2. Microservice architectuur

In dit project is er gekozen voor een Microservice architectuur (zie *Figuur 1 - Microservice architectuur*). Binnen deze architectuur wordt het hele systeem in afzonderlijke componenten opgebouwd. Door het systeem op te splitsen in componenten wordt de totale complexiteit verdeeld in minder complexe componenten met ieder een eigen verantwoordelijkheid. Daarnaast kan voor ieder component een andere implementatie en dataopslag gekozen worden die voor die verantwoordelijkheid het beste is.

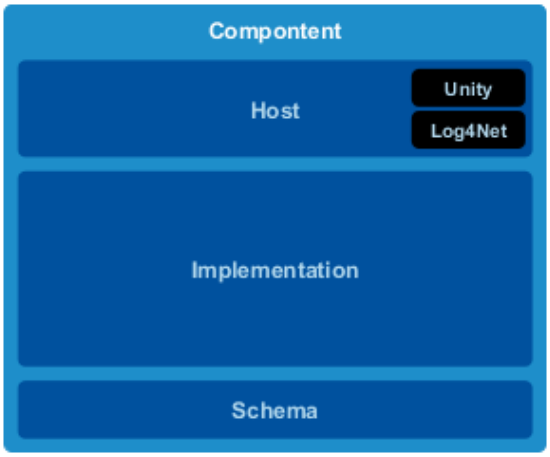


Figuur 1 - Microservice architectuur

9.4 Bijlage D - Technisch Ontwerp

2.1 Componenten

Alle componenten met uitzondering van het dashboard zijn opgebouwd uit drie delen (zie *Figuur 2 - Microservice component architectuur*) waarbij ieder deel resulteert in een eigen DLL. Het voordeel aan deze architectuur is dat het de mogelijkheid bied snel te kunnen wisselen tussen de manier waarop componenten aangeboden wordt zonder dat de implementatie van componenten aangepast moet worden.



Figuur 2 - Microservice component architectuur

Het host deel is verantwoordelijk voor het aanbieden van de beschikbare functionalitieten van het implementatie deel. Dit gebeurt in het geval van de Gateway API in de vorm van een ASP.NET Web API applicatie en in het geval van de metingen service als console applicatie.

Het implementatie deel verzorgt de gewenste functionaliteit en is intern ook opgedeeld in tenminste twee (een boundry- en control laag) of meerdere lagen.

In het schema deel staan alle (publieke) objecten beschreven die door de implementatie gebruikt of aangeboden worden. Deze objecten worden middels de sfutil tool van Info Support gegeneerd op basis van de XSD's over het compontent en over de gebruikte andere compontenten.

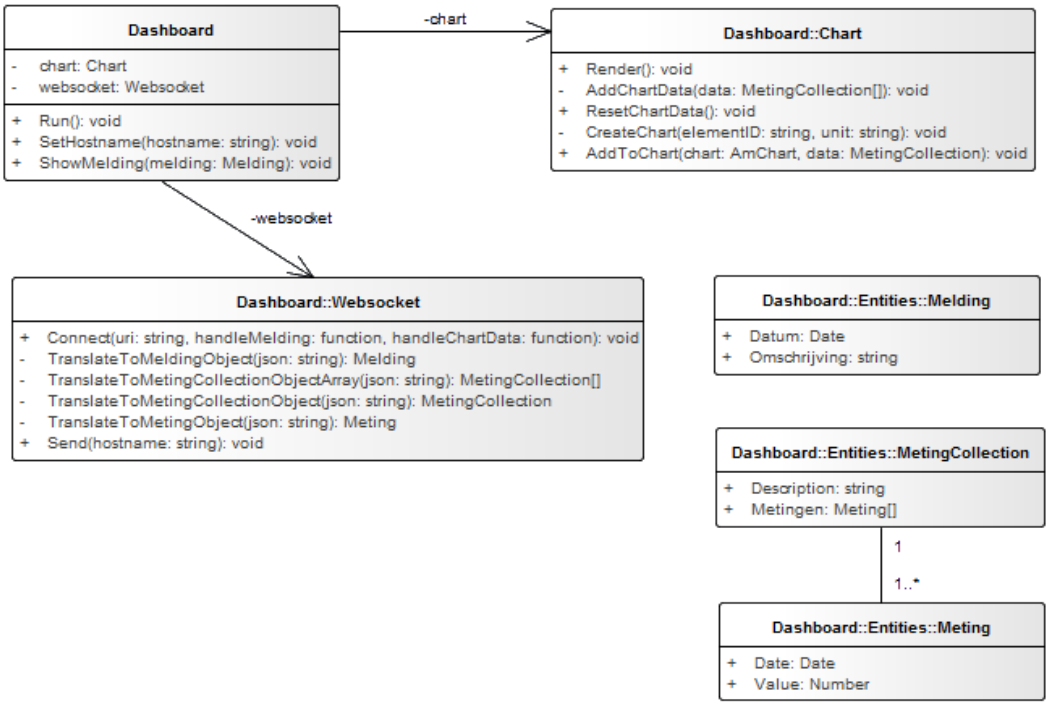
Unity en Log4Net zijn bestaande componenten die in iedere host gebruikt worden. Unity is een framework dat dependency injection mogelijk maakt en Log4Net is een component dat diversen vormen van logging mogelijk maakt.

2.2 Dashboard

Dit is een web applicatie die verantwoordelijk is voor het aanbieden en visualiseren van gegevens aan medewerkers. Met deze applicatie kan de staat en logberichten van een omgeving opgevraagd en getoond worden.

2.2.1 User stories

2.2.2 Klassendiagram



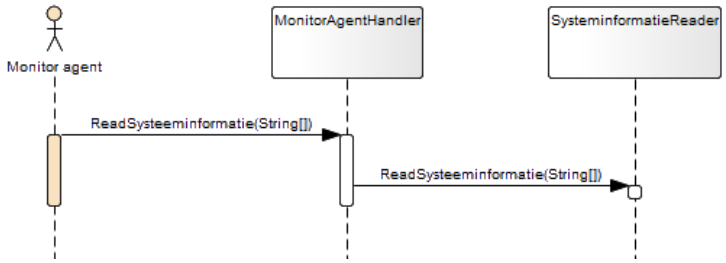
Figuur 3 - Klassendiagram "Dashboard"

2.3 Monitor Agent

Dit is een background applicatie die verantwoordelijk is voor het monitoren van een omgeving. De verkregen logberichten en metingen wordt met een interval van één seconde verstuurd naar het systeem.

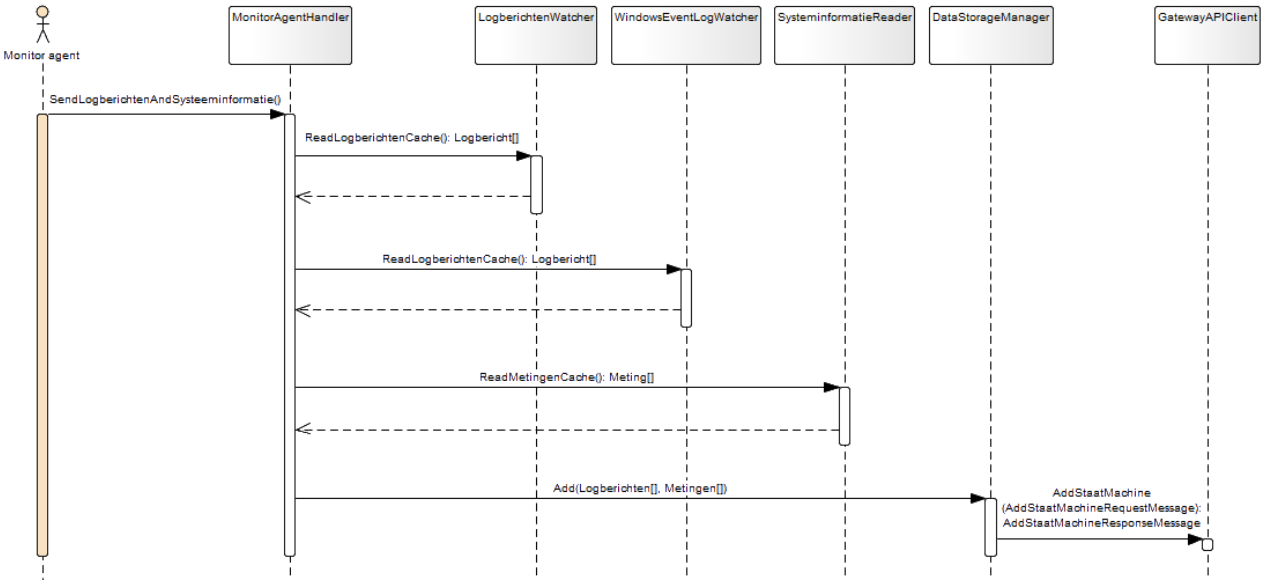
2.3.1 User stories

2.3.1.1 Verkrijgen van systeeminformatie



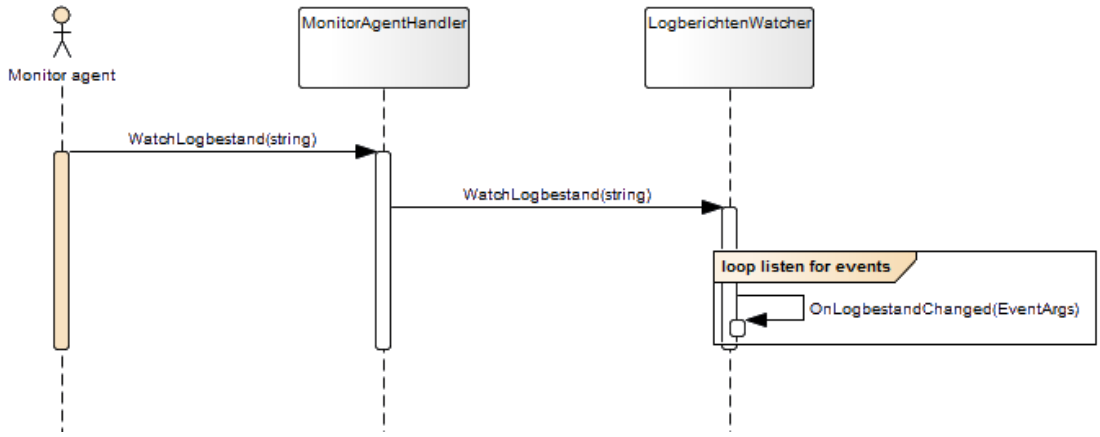
Figuur 4 - Sequence Diagram "UC01 - Verkrijgen van systeeminformatie"

2.3.1.2 Versturen van logberichten & systeeminformatie



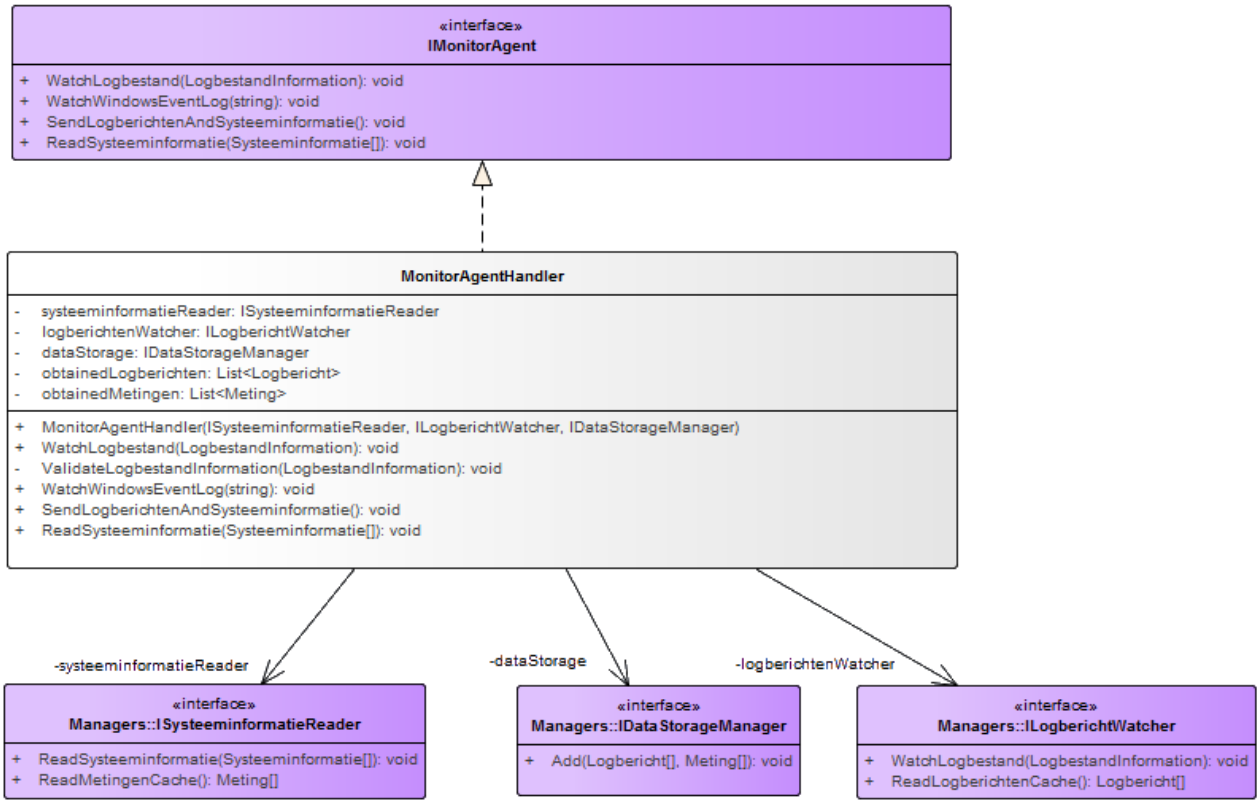
Figuur 5 - Sequence Diagram "UC02 - Versturen van systeem-, beschikbaarheids- en reactie informatie" & "UC06 - Versturen van logberichten"

2.3.1.3 Monitoren van logbestanden



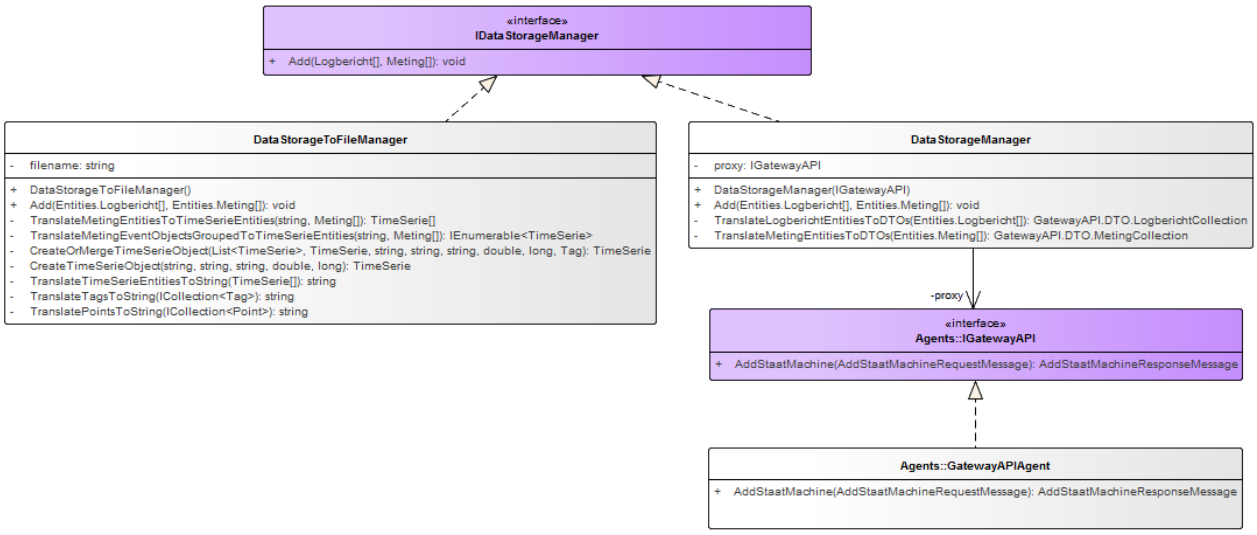
Figuur 6 - Sequence Diagram "UC05 - Monitoren van logbestanden"

2.3.2 Klassendiagram

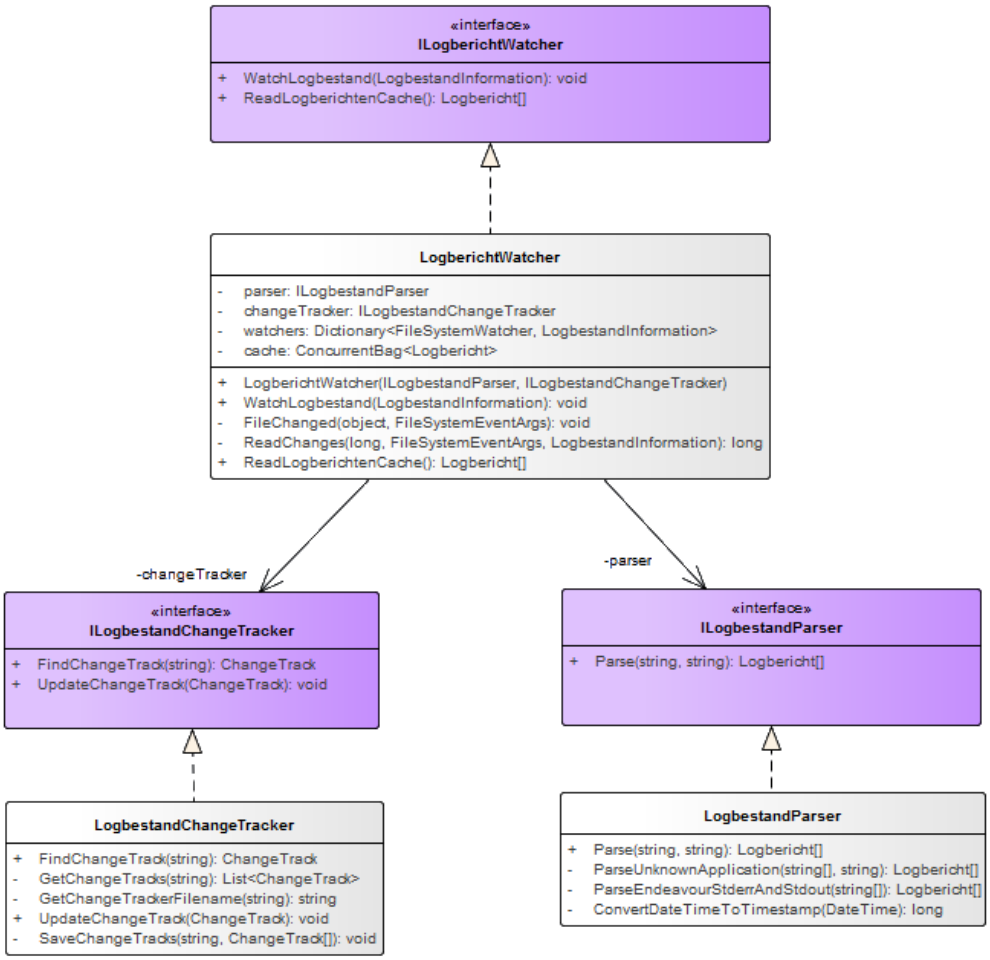


Figuur 7 - Klassendiagram "Monitor Agent implementatie"

9.4 Bijlage D - Technisch Ontwerp

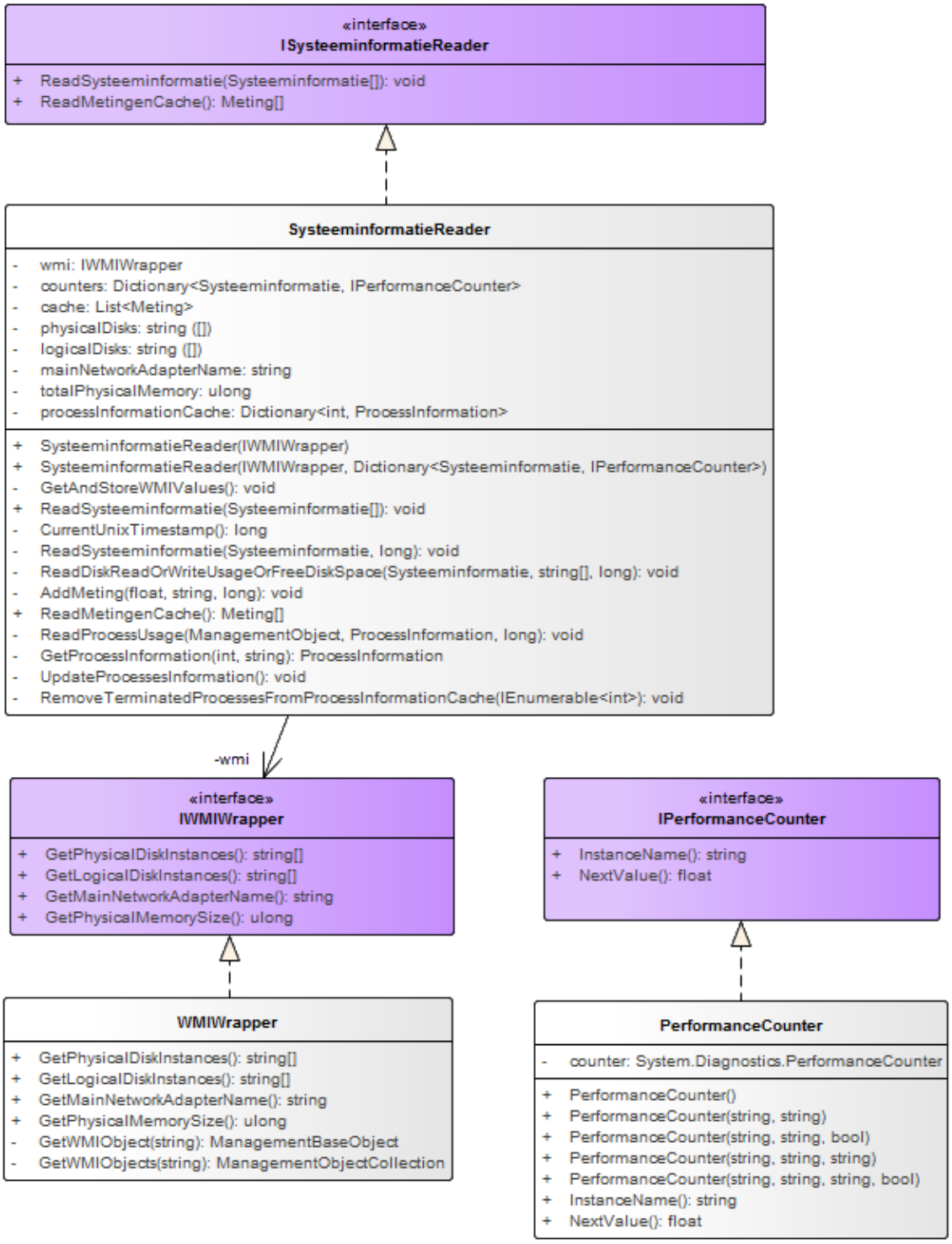


Figuur 8 - Klassendiagram "Monitor Agent Data Storage"



Figuur 9 - Klassendiagram "Monitor Agent logbestanden"

9.4 Bijlage D - Technisch Ontwerp



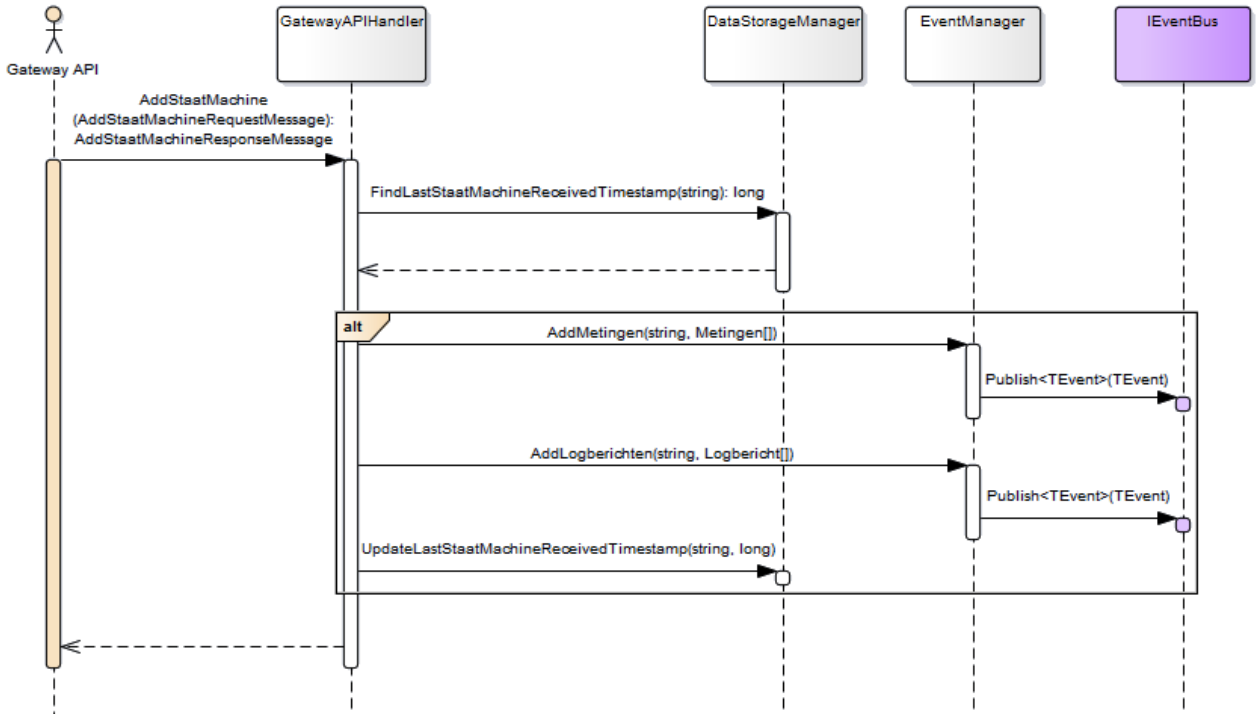
Figuur 10 - Klassendiagram "Monitor Agent systeeminformatie"

2.4 Gateway API

Dit is een web service die verantwoordelijk is voor het aanbieden van een aantal functionaliteiten van het systeem via een REST API. Dit is een component dat veel in Microservice architecturen gebruikt wordt om de complexiteit van aanroepende componenten te verlagen. Aanroepende componenten hoeven geen kennis te hebben van de beschikbare microservices en de (externe) toegang kan beveiligd worden.

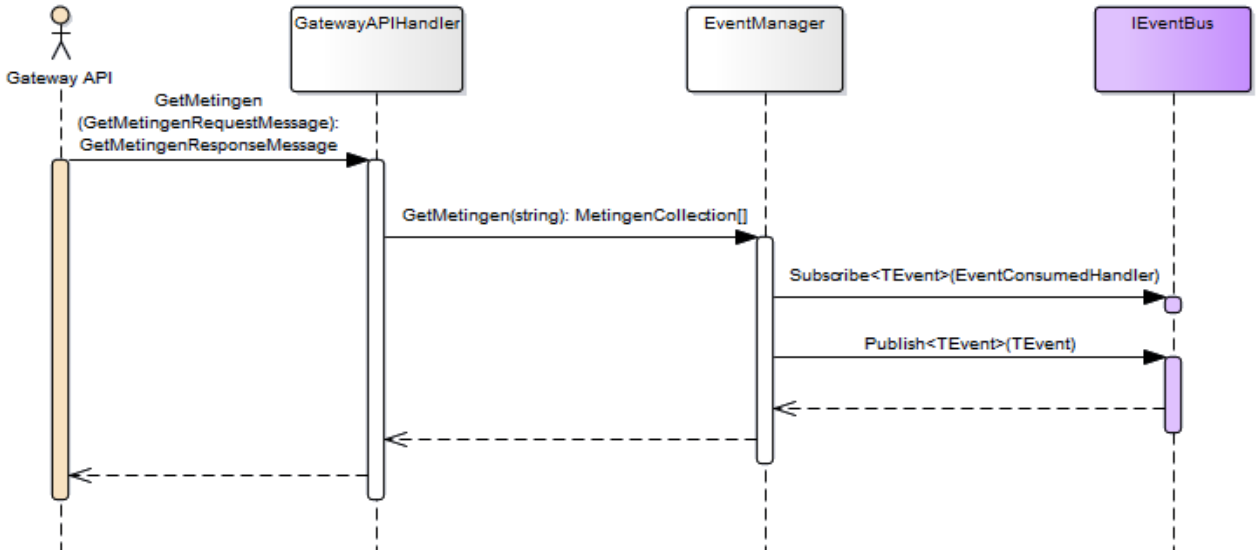
2.4.1 User Stories

2.4.1.1 Versturen van logberichten & systeeminformatie



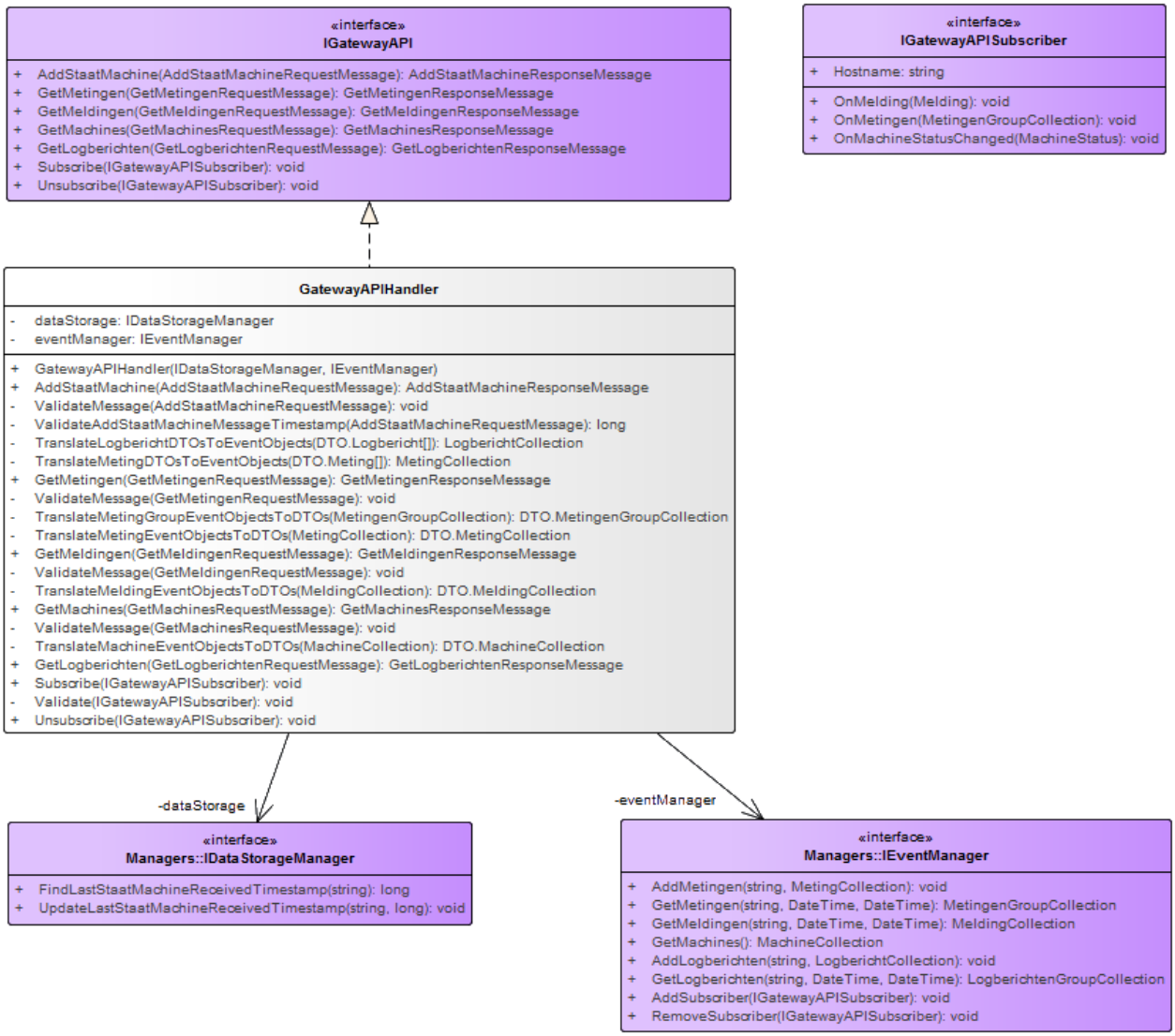
Figuur 11 - Sequence Diagram "UC02 - Versturen van systeem-, beschikbaarheids- en reactie informatie" & "UC06 - Versturen van logberichten"

2.4.1.2 Bekijken van de staat van een machine



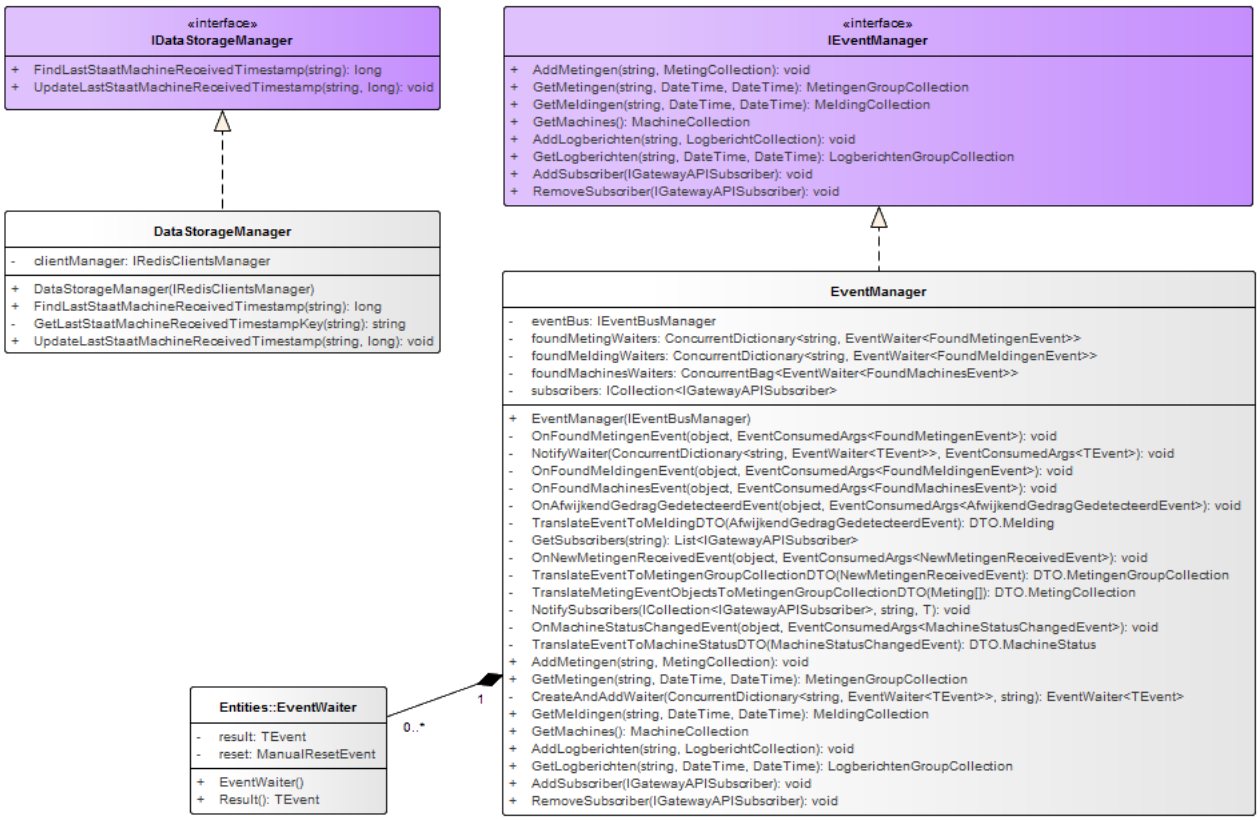
Figuur 12 - Sequence Diagram "UC04 - Bekijken van de staat van een machine"

2.4.2 Klassendiagram



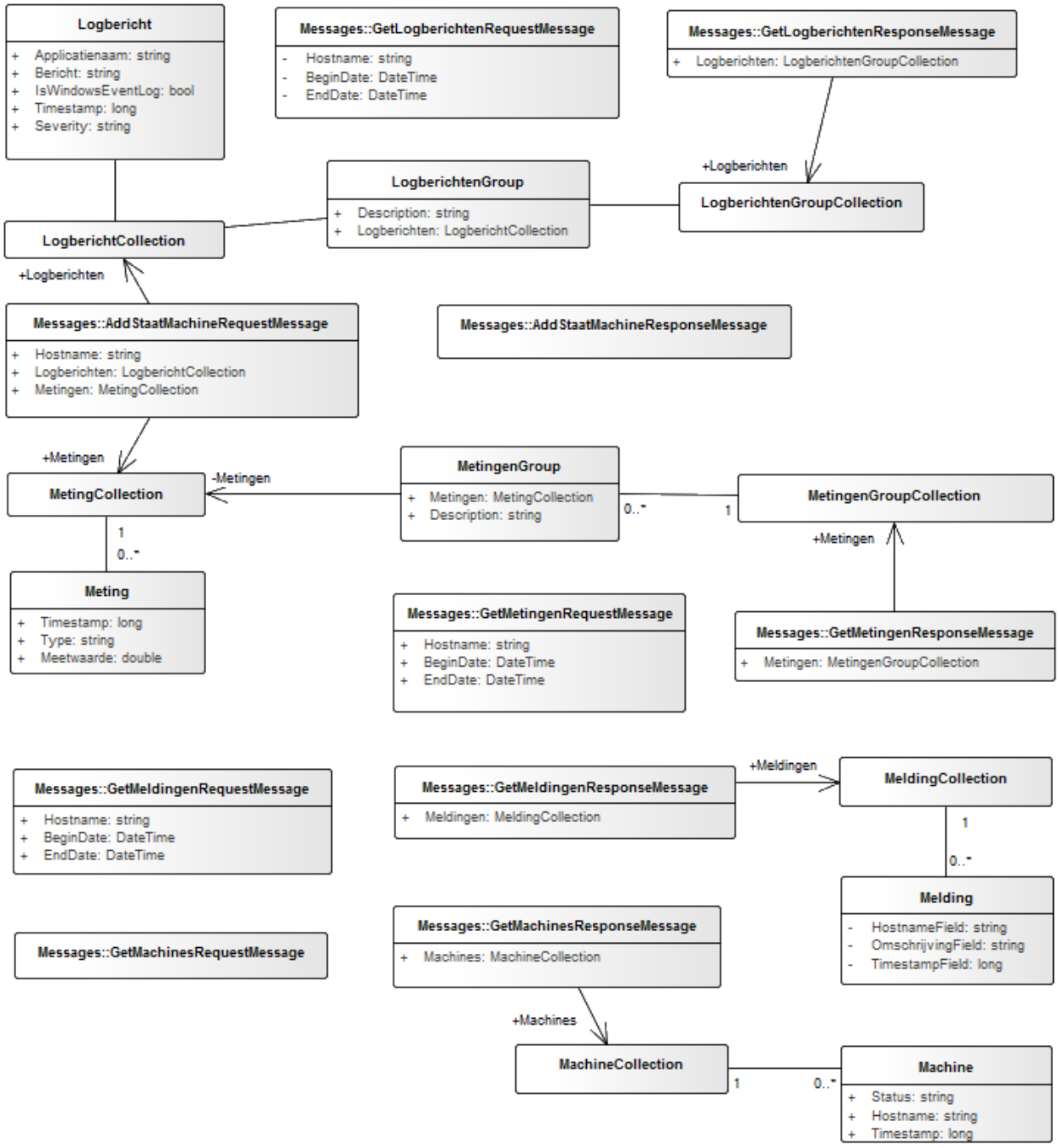
Figuur 13 - Klassendiagram "Gateway API implementatie"

9.4 Bijlage D - Technisch Ontwerp



Figuur 14 - Klassendiagram "Gateway API managers"

9.4 Bijlage D - Technisch Ontwerp



Figuur 15 - Klassendiagram "Gateway API Data Transfer Objects"

2.5 Logberichten Service

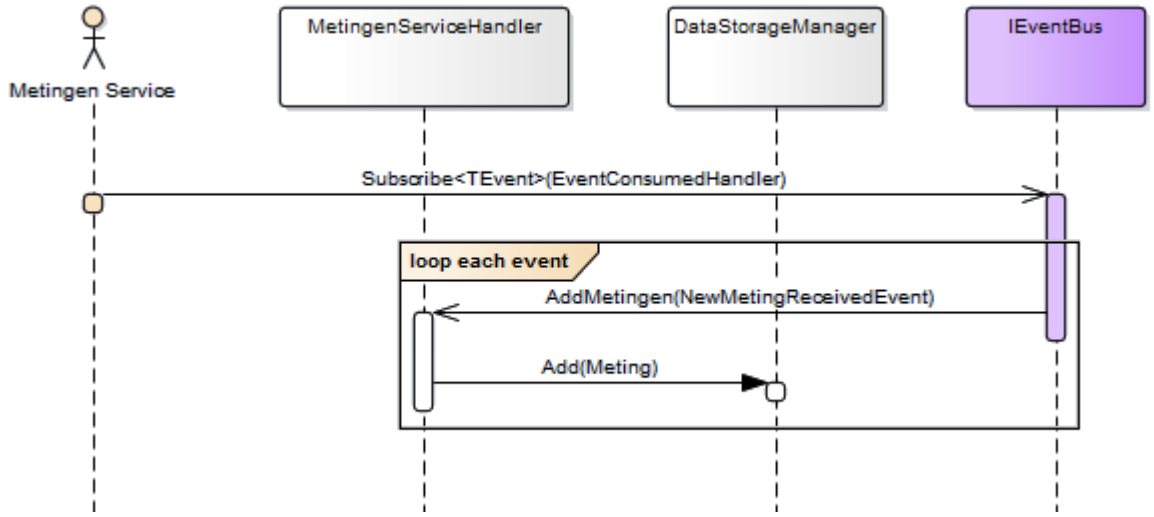
Dit is een service die verantwoordelijk is voor het beheren van logbericht gegevens. Deze service reageert op evenementen die betrekking hebben tot het toevoegen of het opvragen van logberichten.

2.6 Metingen Service

Dit is een service die verantwoordelijk is voor het beheren van meet gegevens. Deze service reageert op evenementen die betrekking hebben tot het toevoegen of het opvragen van metingen.

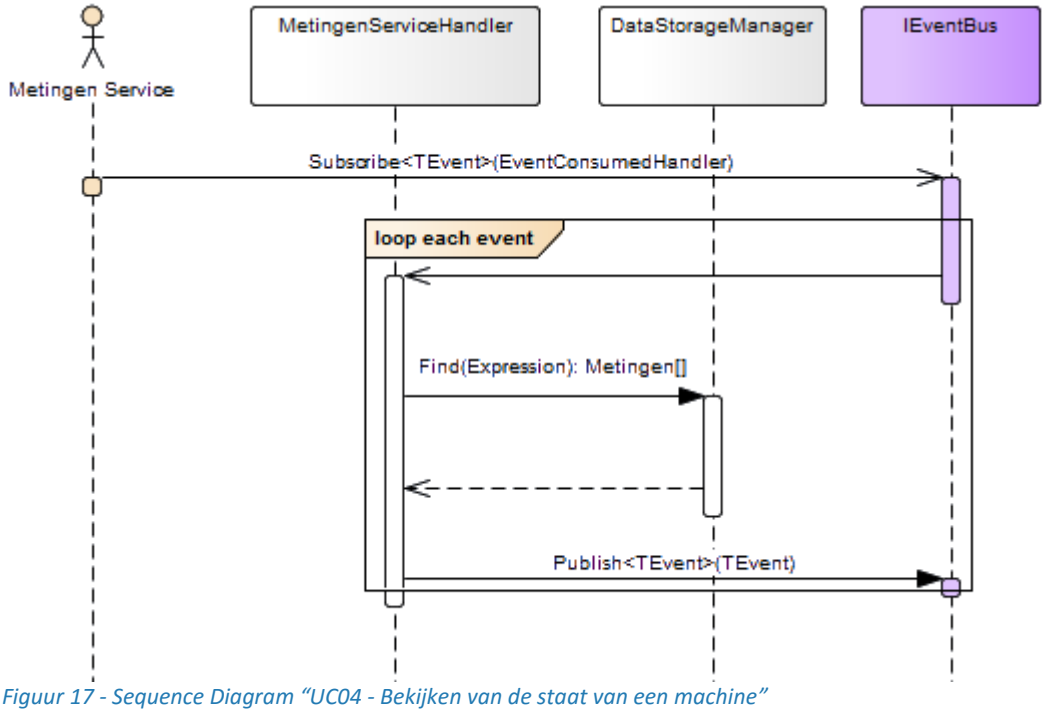
2.6.1 User stories

2.6.1.1 Bewaren metingen



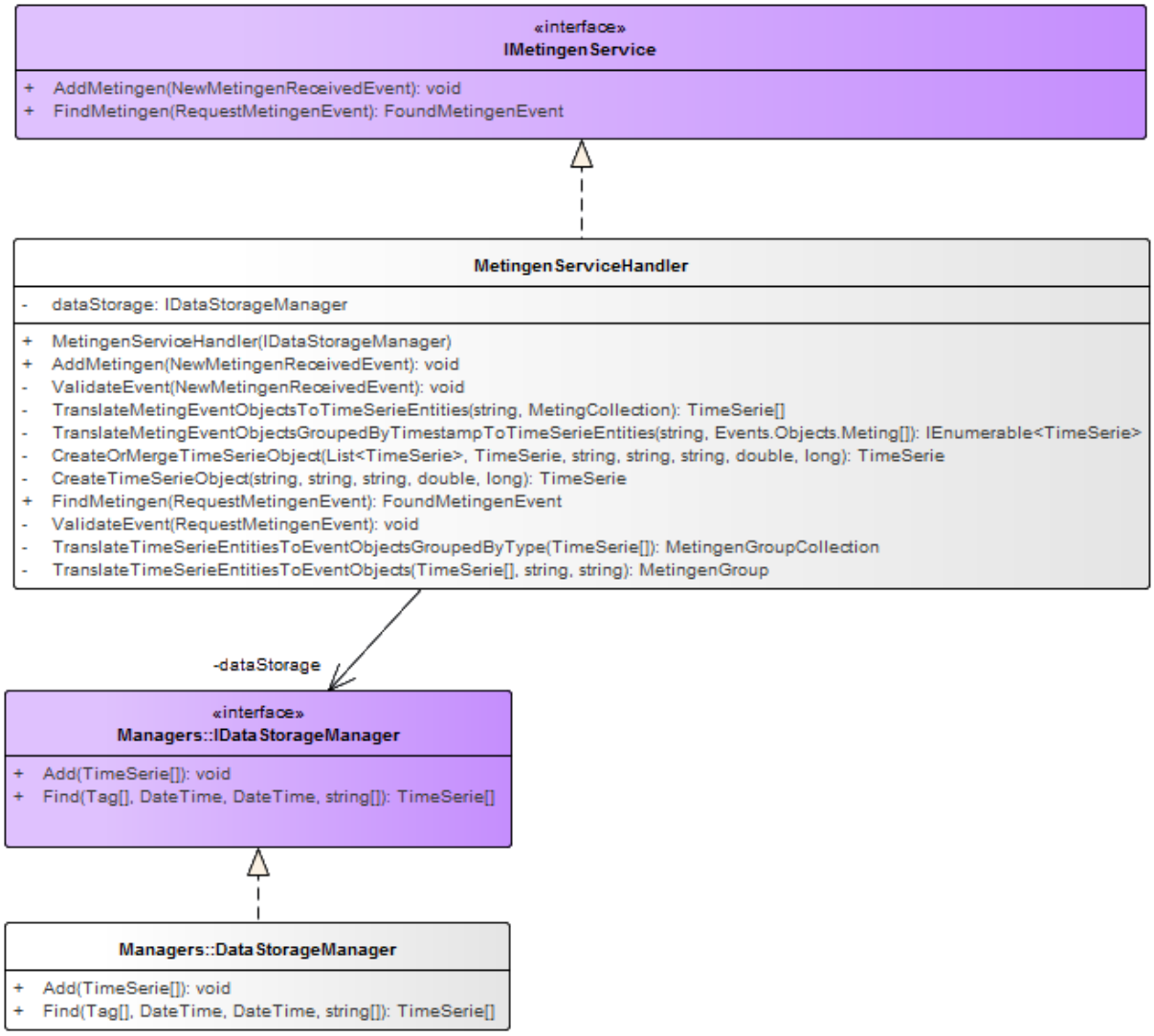
Figuur 16 - Sequence Diagram "UC03 - Bewaren van systeem-, beschikbaarheids- en reactie informatie"

2.6.1.2 Opvragen metingen

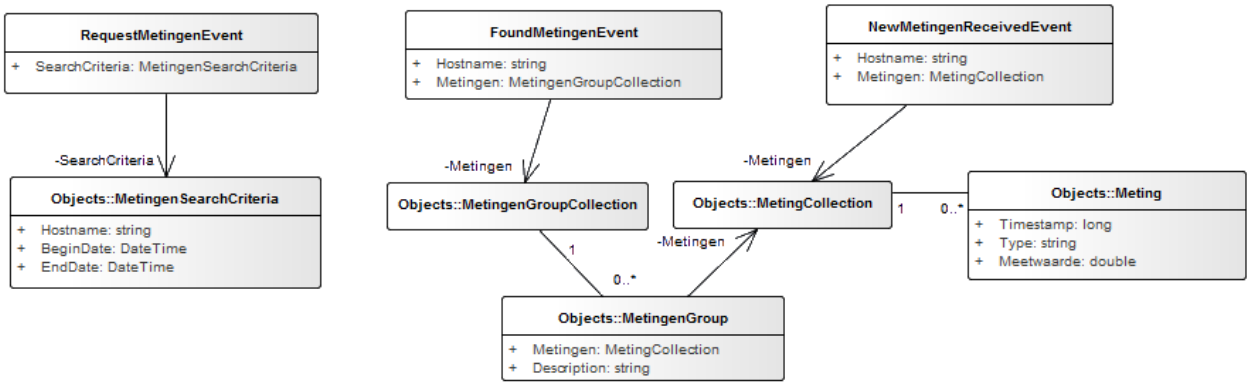


Figuur 17 - Sequence Diagram "UC04 - Bekijken van de staat van een machine"

2.6.2 Klassendiagram



Figuur 18 - Klassendiagram "Metingen Service implementatie"



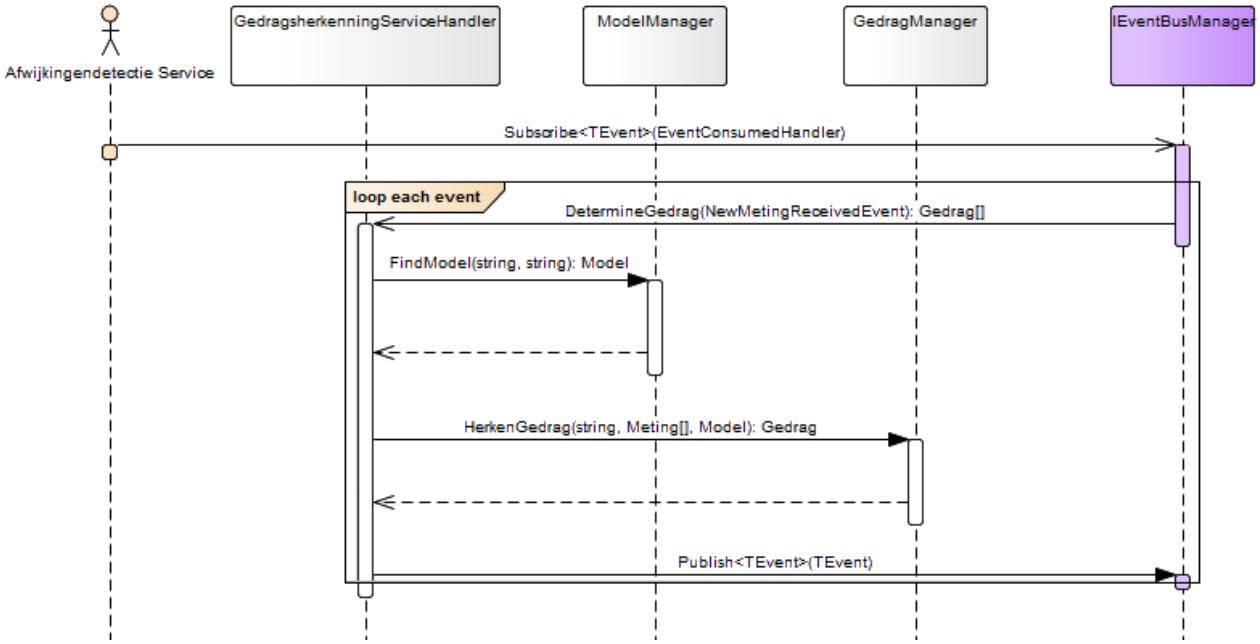
Figuur 19 - Klassendiagram "Metingen Service events"

2.7 Gedragsherkenning Service

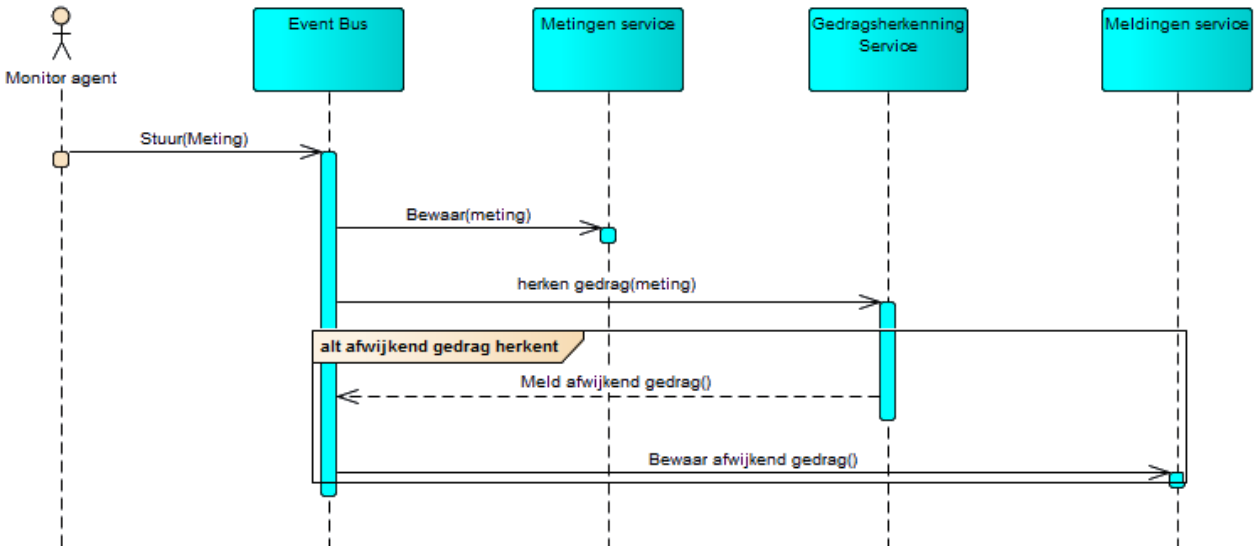
Dit is een service die verantwoordelijk is voor het detecteren van afwijken. Deze service reageert op evenementen die betrekking hebben tot het toevoegen van metingen. Ieder toegevoegde meting wordt gecontroleerd om afwijkend gedrag te herkennen.

2.7.1 User stories

2.7.1.1 Gedrag herkenning

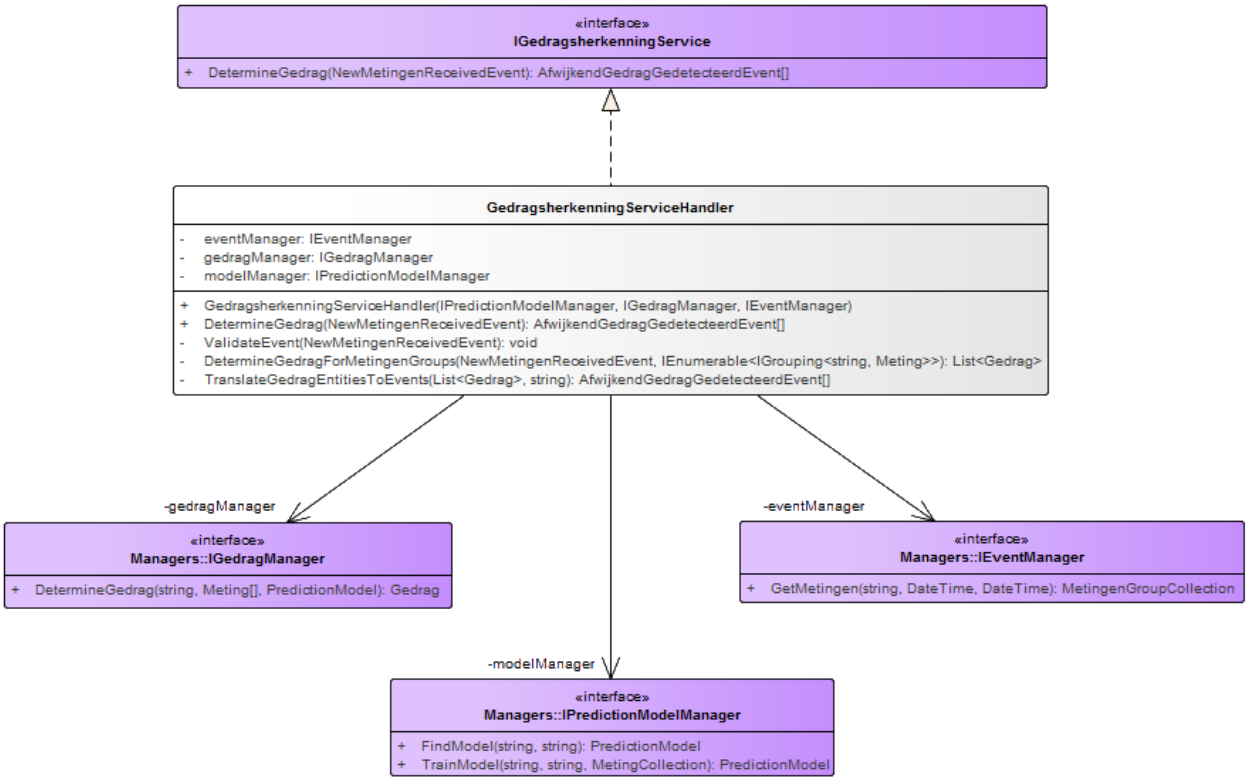


Figuur 20 - Sequence Diagram "UC05 - Voorspellingen aan de hand van systeeminformatie"

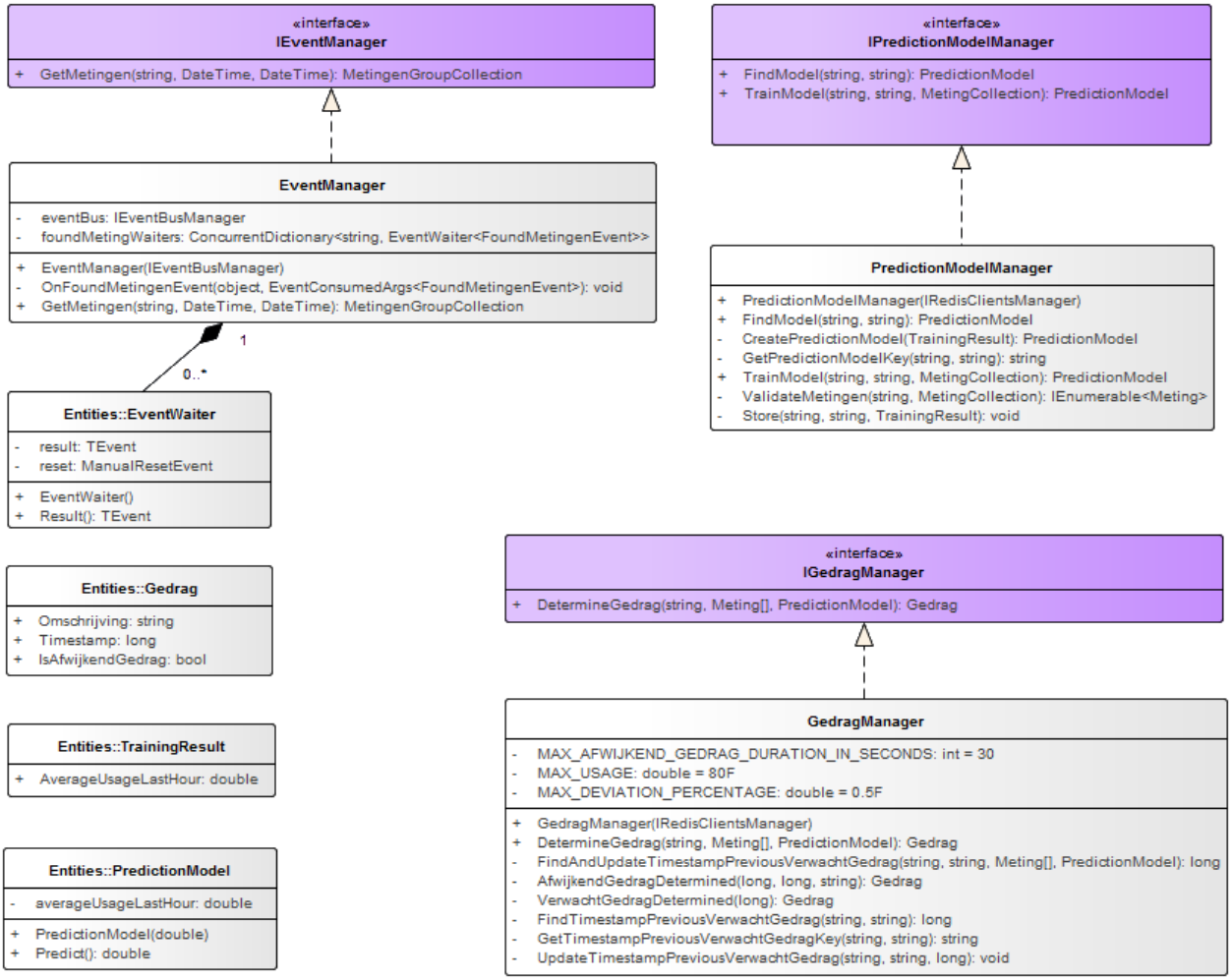


Figuur 21 - Sequence Diagram "UC05 - Voorspellingen aan de hand van systeeminformatie" globaal

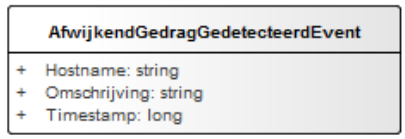
2.7.2 Klassendiagram



Figuur 22 - Klassendiagram "Gedragsherkenning service implementatie"



Figuur 23 - Klassendiagram "Gedragsherkenning service managers"



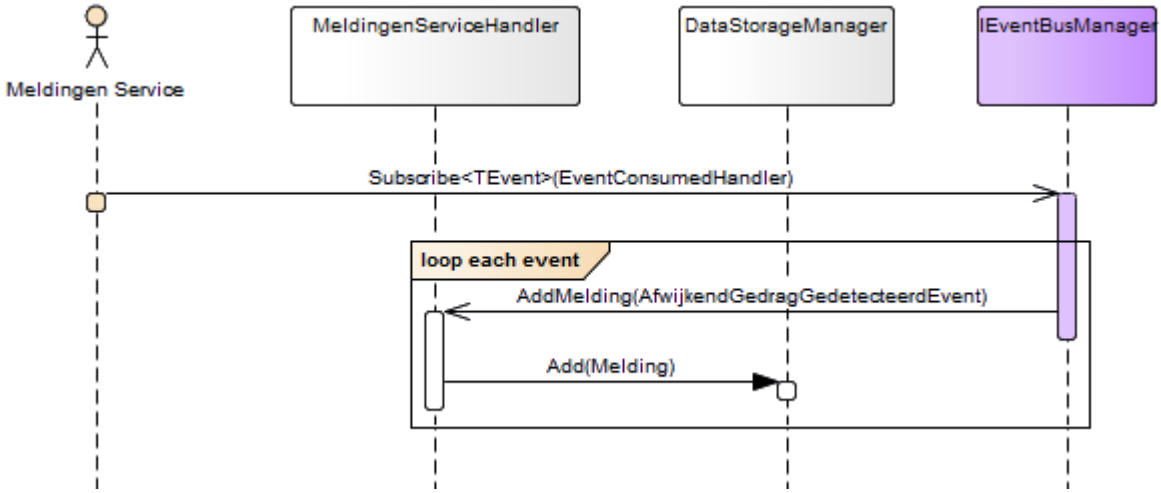
Figuur 24 - Klassendiagram "Gedragsherkenning service event"

2.8 Meldingen Service

Dit is een service die verantwoordelijk is voor het archiveren van gemaakte meldingen. Deze service bewaard gemaakte meldingen zodat deze achteraf opnieuw opgevraagd kunnen worden.

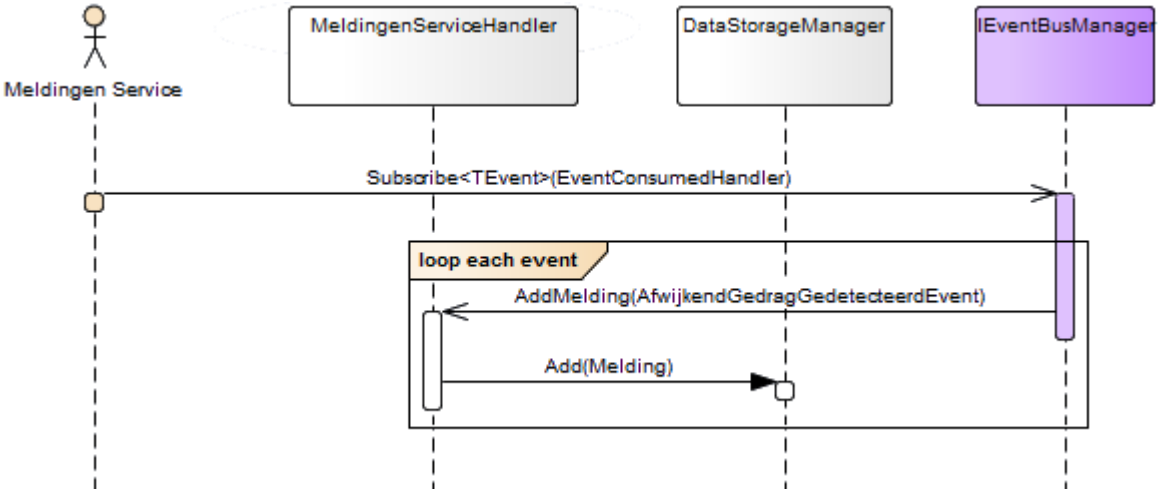
2.8.1 User stories

2.8.1.1 Bewaren melding



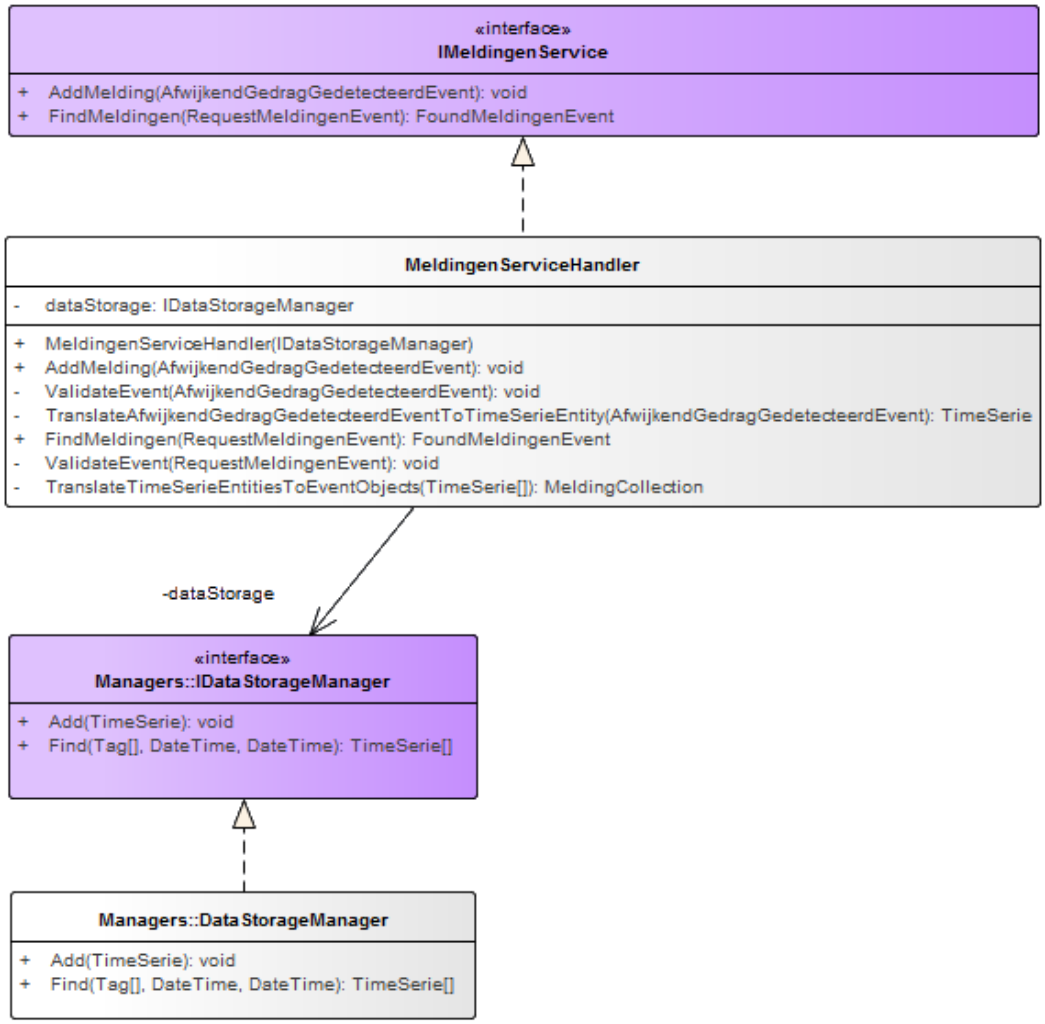
Figuur 25 - Sequence Diagram "UC05 - Voorspellingen aan de hand van systeeminformatie & UC06 - Meldingen bekijken in het dashboard"

2.8.1.2 Opvragen melding

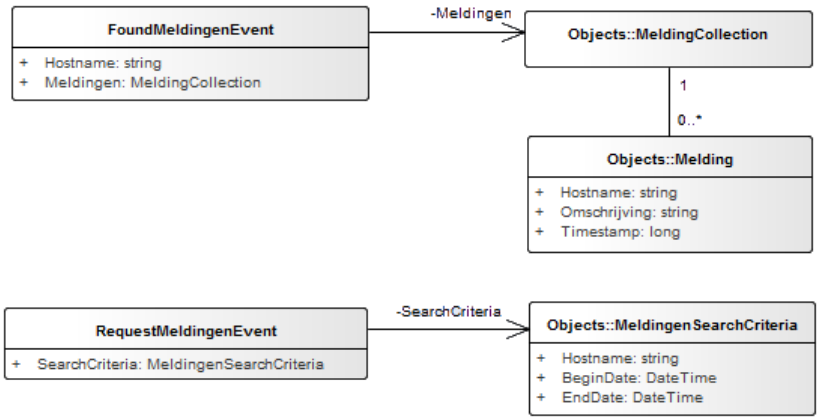


Figuur 26 - Sequence Diagram "UC06 - Meldingen bekijken in het dashboard"

2.8.2 Klassendiagram



Figuur 27 - Klassendiagram "Meldingen service implementatie"



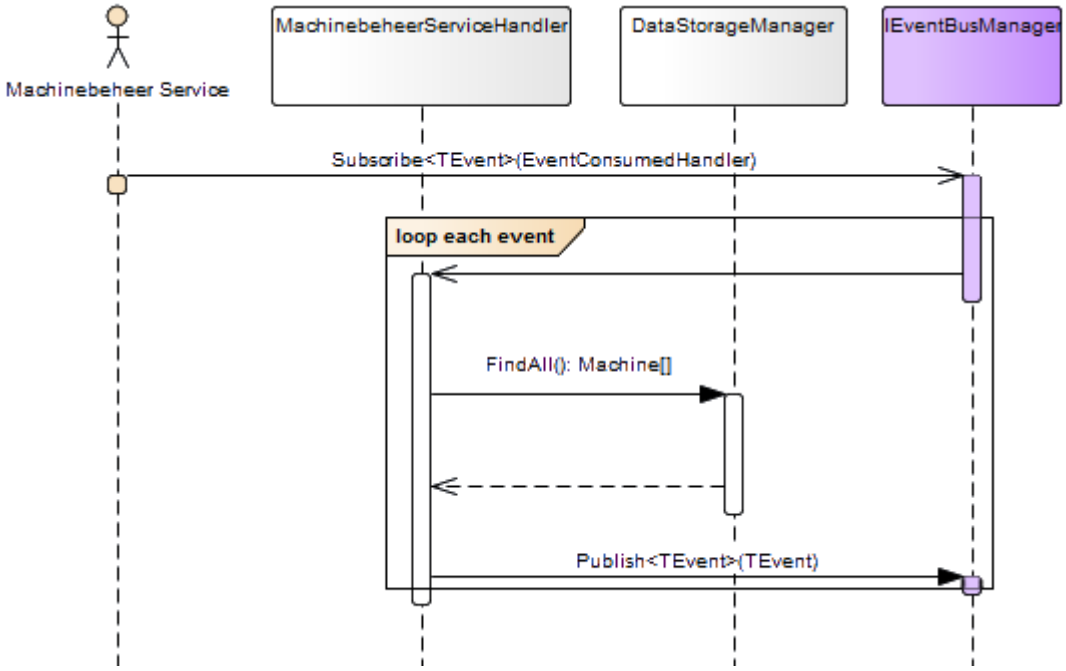
Figuur 28 -Klassendiagram "Meldingen service events"

2.9 Machinebeheer service

Dit is een service die verantwoordelijk is voor het beheren van alle machine informatie. Deze service bewaard alle machine informatie (zoals de host name maar ook de huidige staat) van machines die opgenomen zijn in het systeem.

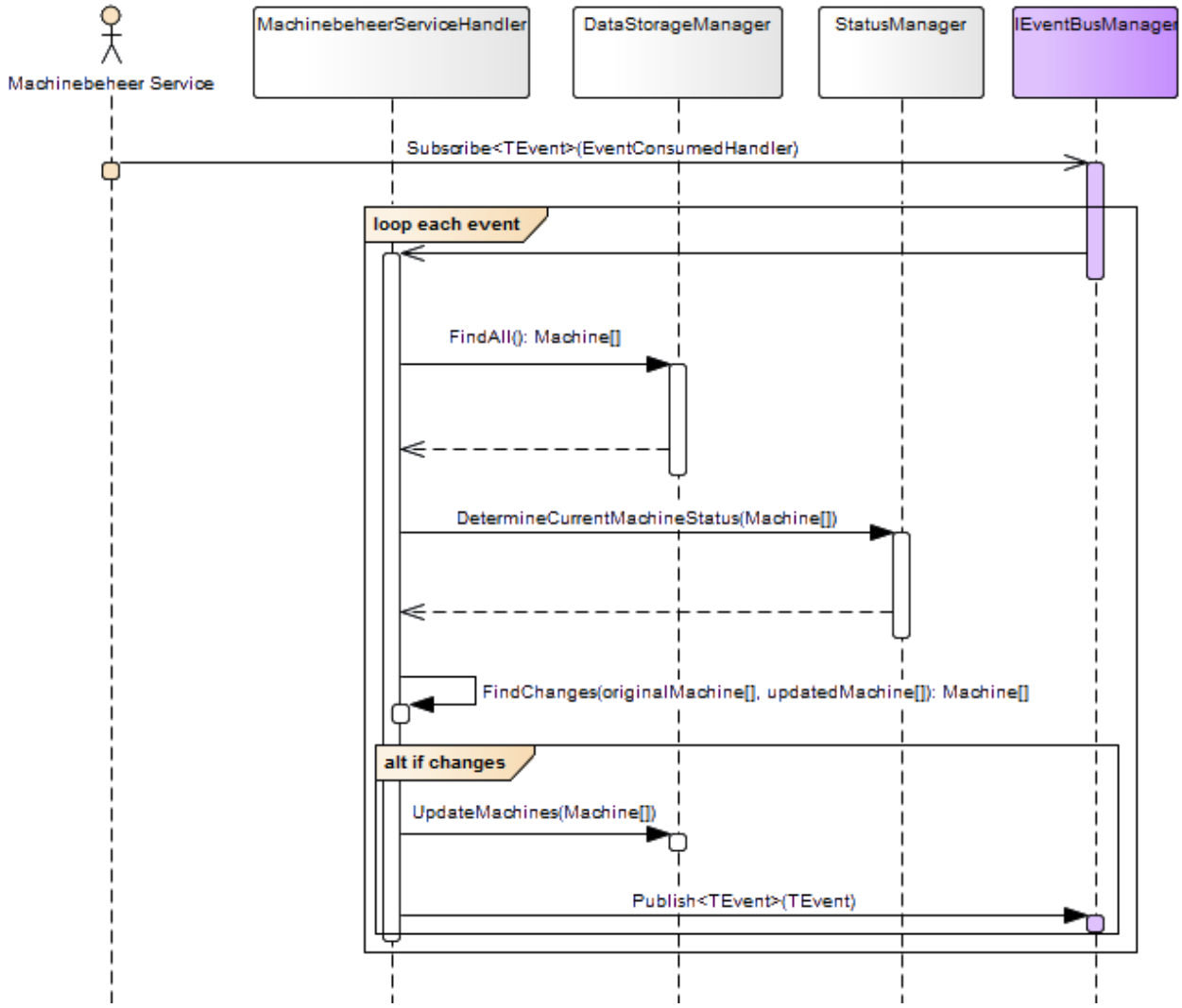
2.9.1 User stories

2.9.1.1 Opvragen machines



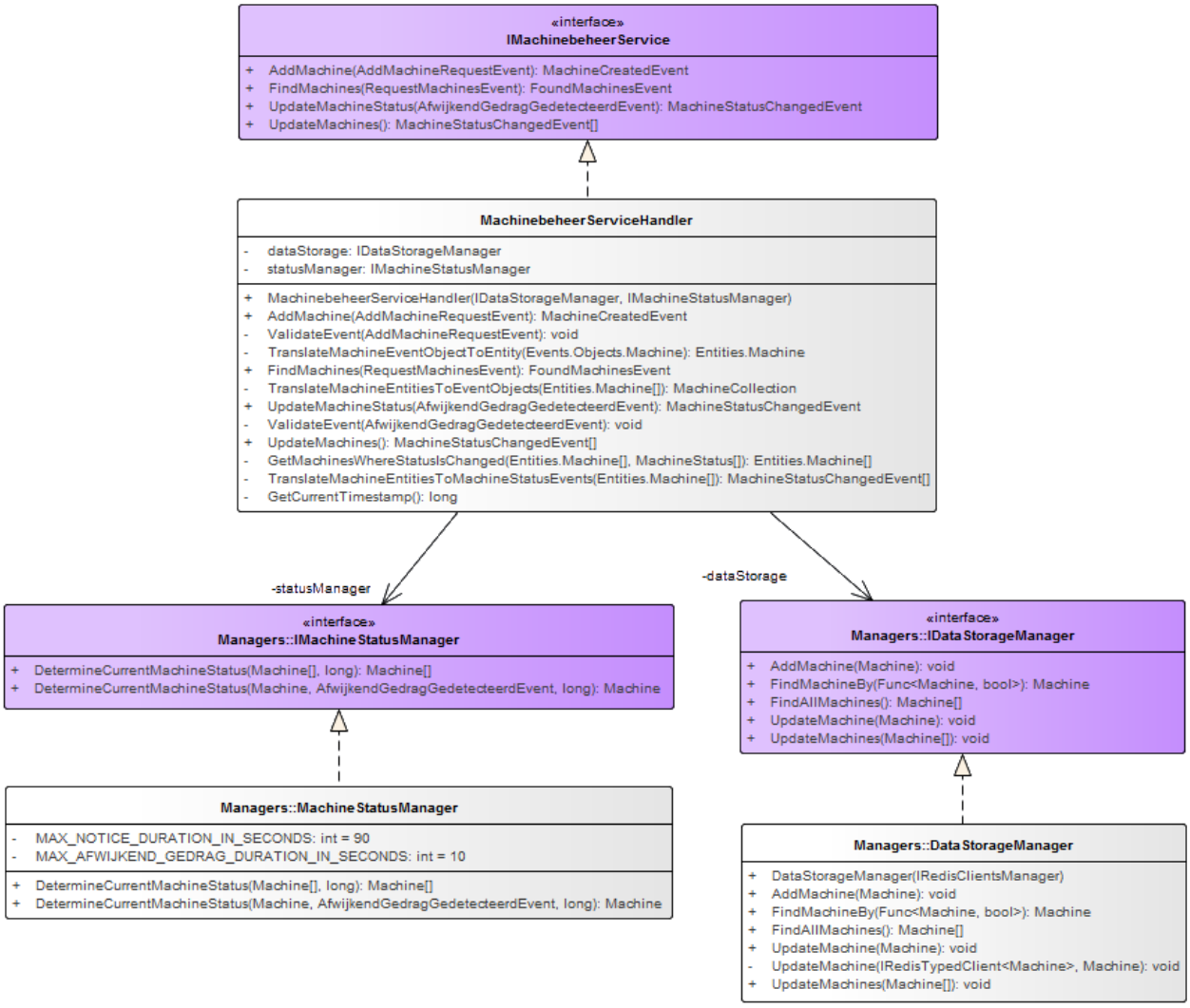
Figuur 29 - Sequence Diagram "UC07 - Bekijken van overzichtsscherm"

2.9.1.2 Bijwerken machinestatus

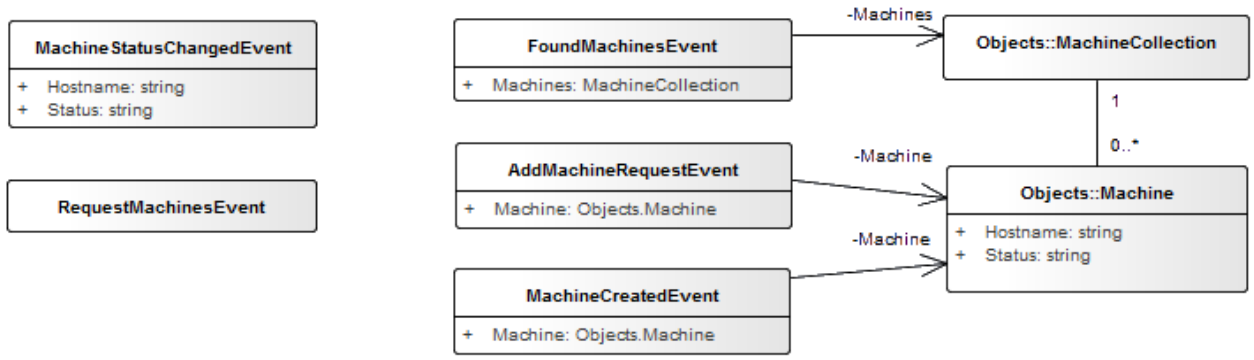


Figuur 30 - Sequence Diagram "UC07 - Bekijken van overzichtsscherm"

2.9.2 Klassendiagram



Figuur 31 - Klassendiagram "Machinebeheer service implementatie"

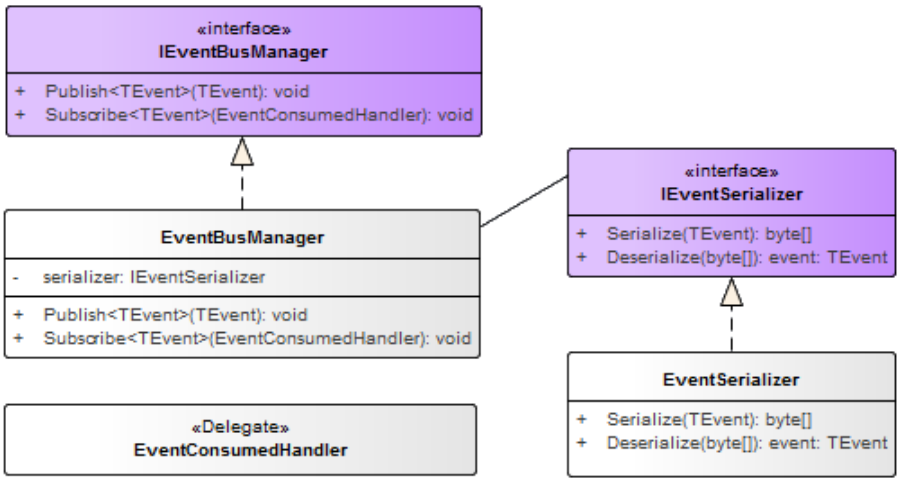


Figuur 32 - Klassendiagram "Machinebeheer service events"

2.10 Event Bus

De event bus is verantwoordelijk voor het voorzien van asynchroon communicatie tussen componenten. De objecten die gepubliceerd worden op de event bus zijn in XSD's gedefinieerd. Alle events in RabbitMQ zijn in JSON.

2.10.1 Klassendiagram



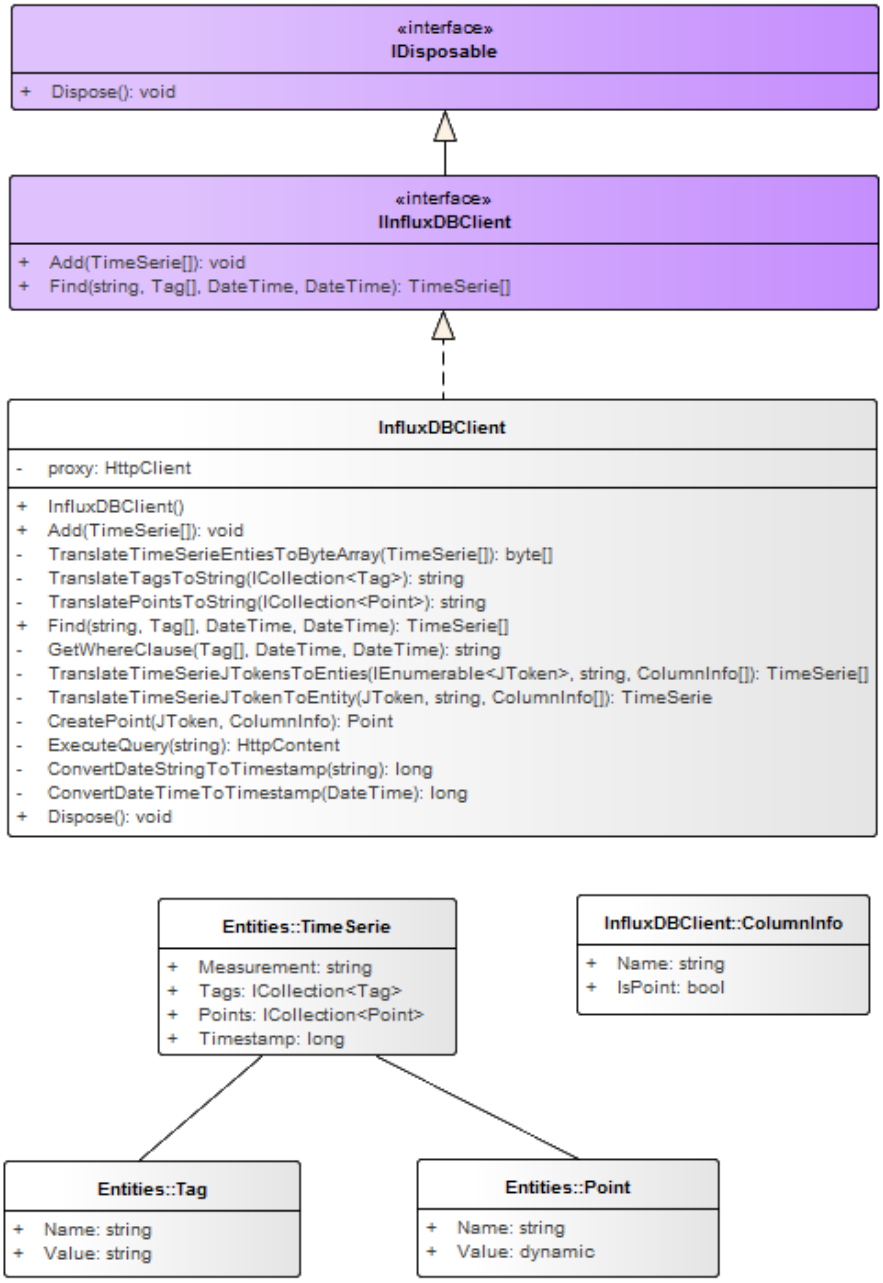
Figuur 33 - Klassendiagram "Event Bus"

2.10.2 Voorbeeld XSD

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="urn:psf:metingenservice:events:objects"
    elementFormDefault="qualified"
    xmlns:tns="urn:psf:metingenservice:events:objects"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
>

      <xs:complexType name="Meting">
            <xs:sequence>
                  <xs:element name="Timestamp" type="xs:long" />
                  <xs:element name="Type" type="xs:string" />
                  <xs:element name="Meetwaarde" type="xs:double" />
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="MetingCollection">
            <xs:sequence>
                  <xs:element name="Meting" type="tns:Meting" minOccurs="0"
                        maxOccurs="unbounded" />
            </xs:sequence>
      </xs:complexType>

</xs:schema>
```

3. InfluxDB .NET connector

Bestaande .NET connectoren voor InfluxDB bleken niet na behoren of zelfs niet te functioneren. Er is daarom een eigen connector ontwikkeld die gebruik maakt van de REST API die InfluxDB aanbied.



Figuur 34 - Klassendiagram InfluxDB .NET connector

Data-analyse

Predicting Server Flaws

Anthony Huebers

Data-analyse

Predicting Server Flaws

Titel	Data-analyse
Project/Onderwerp	Predicting Server Flaws
Versie	1.1
Status	Definitief
Datum	21-dec-2015
Bestand	Data-Analyses - Anthony Huebers
Bedrijf	Info Support B.V.

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

9.5 Bijlage E - Data-analyse

Historie

Versie	Status	Datum	Auteur	Verandering
1.0	Concept	09-12-2015	Anthony Huebers	Creatie
1.1	Definitief	21-12-2015	Anthony Huebers	Feedback Hylke Peek verwerkt

Distributie

Versie	Status	Datum	Aan
1.0	Concept	15-12-2015	Tom Nieuwenhuijs en Hylke Peek

Referenties

Code	Bron

9.5 Bijlage E - Data-analyse

Inhoudsopgave

9.5 Bijlage E - Data-analyse

1. Inleiding

Binnen het project wordt een (proof of concept) systeem gerealiseerd wat (patronen in) problemen kan herkennen en mogelijk problemen in de toekomst kan voorspellen. Dit document beschrijft hoe de data geanalyseerd is en welke resultaten dit opleverde. De resultaten die in dit document beschreven worden zijn gebaseerd op dezelfde dataset afkomstig van één willekeurige machine uit de complete dataset. Er is gekozen voor één dezelfde willekeurige dataset omdat de voorspelling voor alle datasets (hetzelfde) moeten werken. In dit document wordt met “willekeurig” bedoelt dat door middel van “random algoritmes” door de computer keuzes gemaakt worden

In de dataset staan van één week voor iedere seconden de volgende gegevens:

- Het (totale) processor verbruik
- Het (totale) geheugen verbruik
- De dag van de week
- Het uur op de dag
- De datum van de meting (als timestamp sinds 1 januari 1970)
- Of er een logbericht is
- De severity van een logbericht
- De namen van ieder actief (niet door de systeem uitgevoerd) proces
- Het processor verbruik en geheugen verbruik voor ieder actief (niet door de systeem uitgevoerd) proces

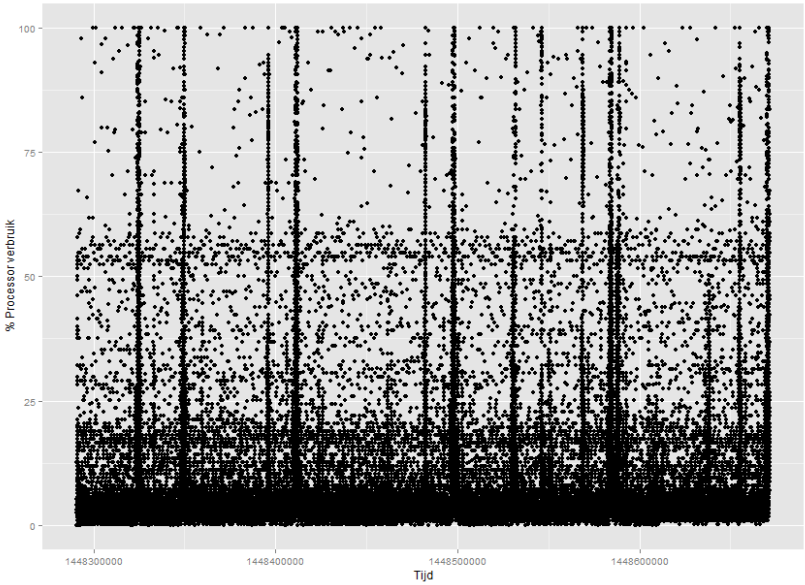
2. Classificatie

De verzamelde metingen zijn niet geclassificeerd. Dit betekend dat voor iedere meting niet bekend is of deze afwijkend gedrag vertoont. Wanneer de metingen geclassificeerd zijn kan een model getraind worden wat door middel van een of meerdere gegevens van een meting voorspellingen over toekomstige metingen kan maken. In overleg met de opdrachtgever zijn de volgende criteria opgesteld als mogelijke classificatie:

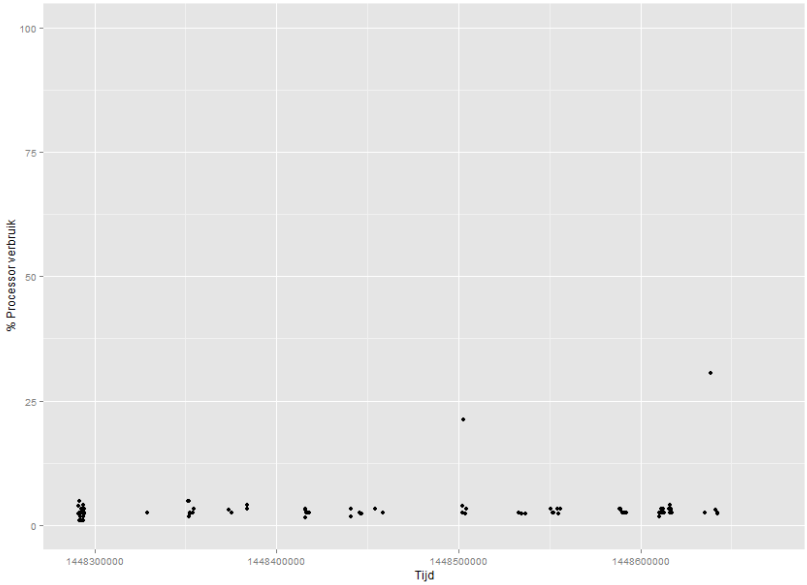
- Het (totale) processor verbruik is boven de 80%
- Het (totale) geheugen verbruik is boven de 80%
- Een foutmelding is geregistreerd met de severity anders dan info of debug

9.5 Bijlage E - Data-analyse

2.1 Analyses



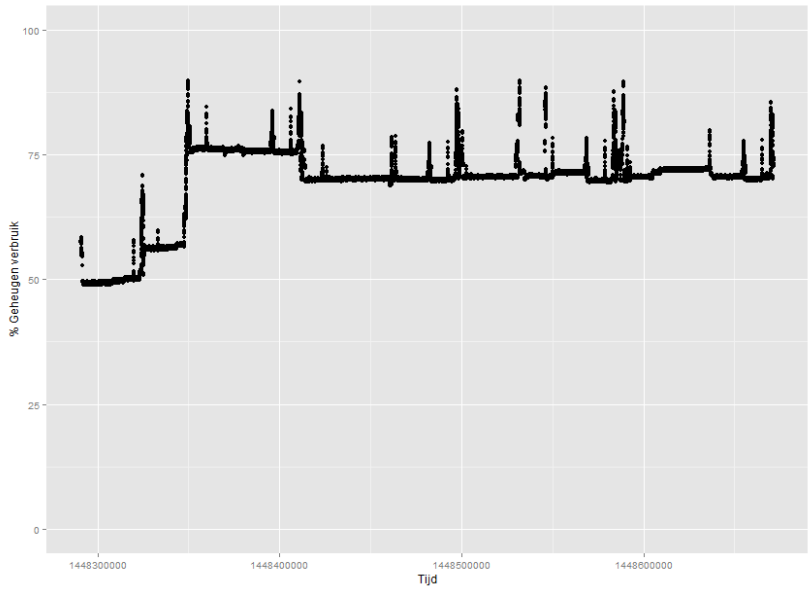
Figuur 1 - Processor verbruik



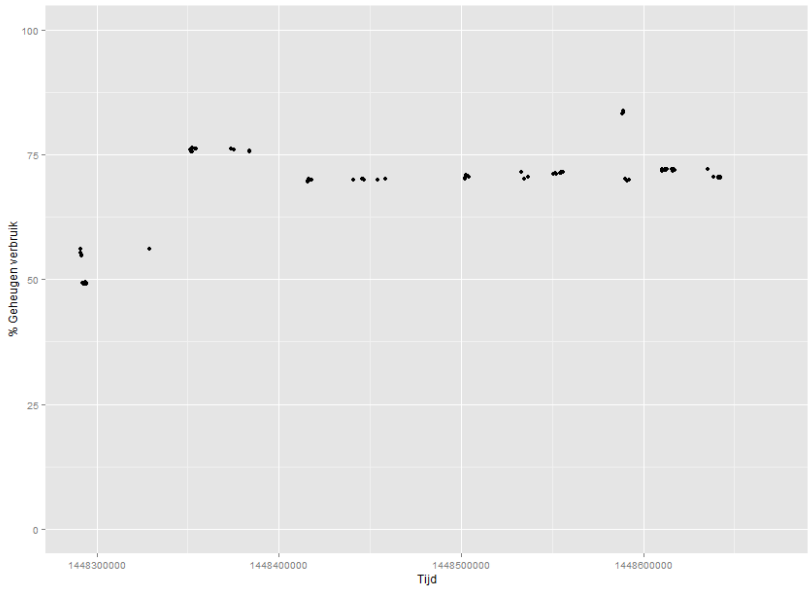
Figuur 2 - Processor verbruik bij een foutmelding

In *Figuur 1 - Processor verbruik* is het (totale) processor verbruik te zien. In *Figuur 2 - Processor verbruik bij een foutmelding* is het (totale) processor verbruik te zien op het moment dat een foutmeldingen is geregistreerd met de severity anders dan info of debug. Wat voornamelijk opvalt aan deze figuren is dat het processor gebruik op de momenten dat foutmeldingen worden geregistreerd niet uitschiet naar een heel hoog of laag verbruik maar continue blijft.

9.5 Bijlage E - Data-analyse



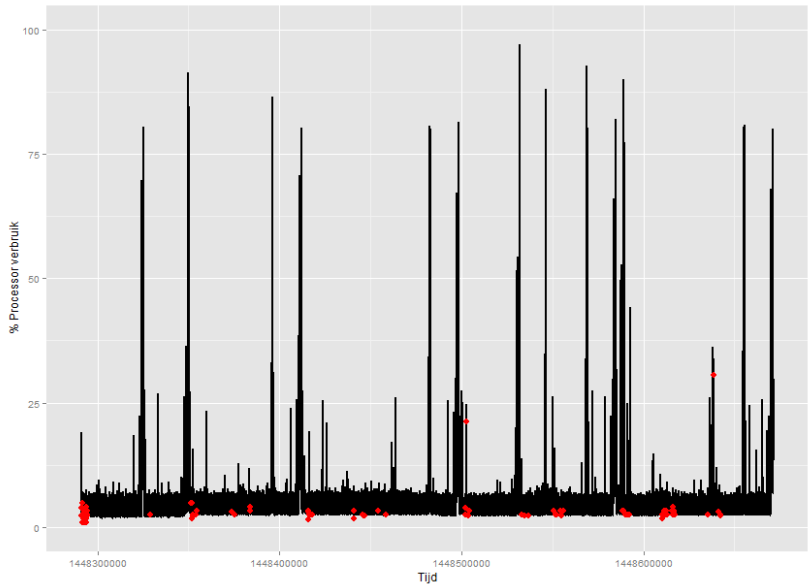
Figuur 3 - Geheugen verbruik



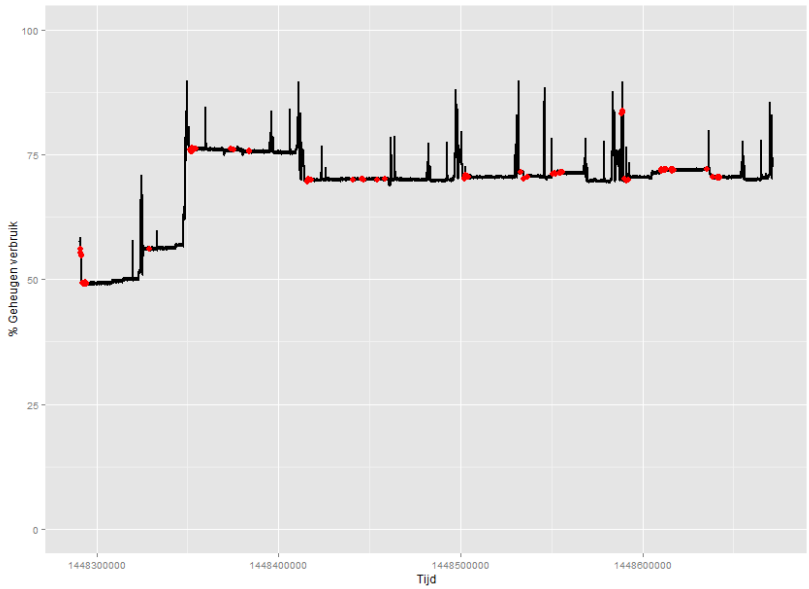
Figuur 4 - Geheugen verbruik bij een foutmelding

In *Figuur 3 - Geheugen verbruik* is het (totale) geheugen verbruik te zien. In *Figuur 4 - Geheugen verbruik bij een foutmelding* is het (totale) geheugen verbruik te zien op het moment dat een foutmeldingen is geregistreerd met de severity anders dan info of debug. Wat voornamelijk opvalt aan deze figuren is dat het geheugen gebruik op de momenten dat foutmeldingen worden geregistreerd ongeveer het zelfde blijft.

9.5 Bijlage E - Data-analyse



Figuur 5 - Voortschrijdend gemiddelde processor verbruik gecombineerd met het processor verbruik bij een foutmelding



Figuur 6 - Geheugen verbruik gecombineerd met het geheugen verbruik bij een foutmelding

In *Figuur 5 - Voortschrijdend gemiddelde processor verbruik gecombineerd met het processor verbruik bij een foutmelding* is het voortschrijdend gemiddelde (totale) processor verbruik gecombineerd met het (totale) processor verbruik te zien op het moment dat een foutmeldingen is geregistreerd (rode punt) met de severity anders dan info of debug. Door het voortschrijdend gemiddelde (over 30 seconden) te gebruiken wordt de fluctuaties in processor verbruik minder. In *Figuur 6 - Geheugen verbruik gecombineerd met het geheugen verbruik bij een foutmelding* is het (totale) geheugen verbruik gecombineerd met het (totale) geheugen verbruik te zien op het moment dat er een foutmeldingen is geregistreerd (rode punt) met de severity anders dan info of debug.

2.2 Classificatie algoritmes

Aan de hand van de onderstaande criteria is de dataset geclassificeerd als afwijkend gedrag:

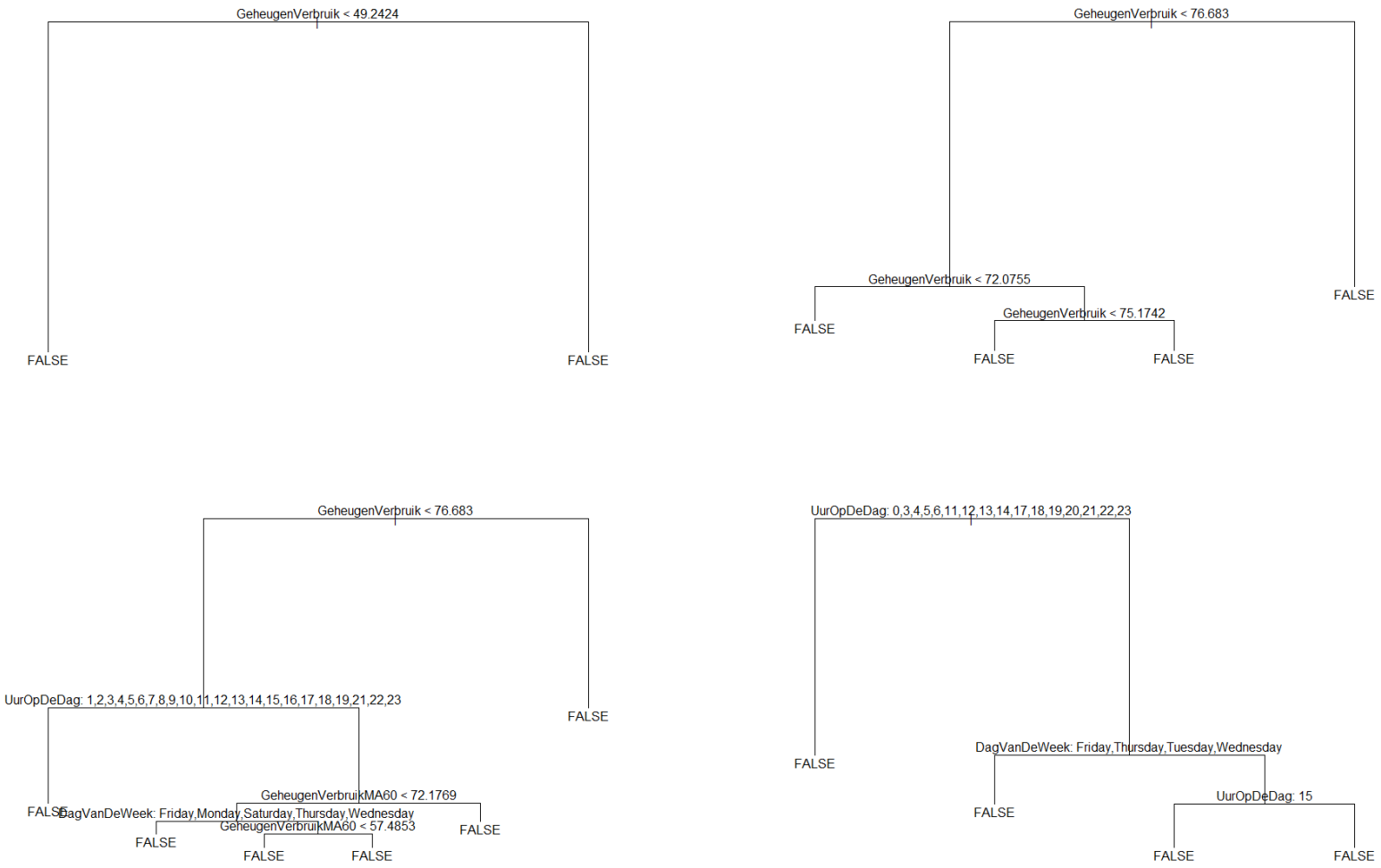
- Het (totale) processor verbruik is boven de 80%
- Het (totale) geheugen verbruik is boven de 80%
- Een foutmelding is geregistreerd met de severity anders dan info of debug

Bij het maken van het model is 70% van de dataset gebruikt om het model te trainen. De resterende 30% is gebruikt om het gemaakte model te testen. De verdeling van deze dataset is willekeurig gebeurt en niet aan de hand van de volgorde (dus trainging dataset is niet per definitie gelijk aan de eerste 70% gegevens).

Na het classificeren van de dataset (per criteria en alle criteria gecombineerd) zijn meerdere classificatie algoritmes toegepast op de dataset. Hiervoor zijn een aantal veel benoemde classificatie algoritmes gebruikt *(Wesley, 2013)* (Quick-R, z.d.). De resultaten wisselde per algoritme maar alle algoritmes voorspelde alles fout. In *Figuur 7 - Decisiontree* zijn een aantal modellen te zien. In alle modellen is te zien dat alle paden resulteren in FALSE, wat wil zeggen dat het geen afwijkend gedrag is.

De correctheid van deze modellen was rond de 99%. Dit was te verklaren omdat rond de 99% van de gegevens gemarkeerd was als niet afwijkend gedrag. Omdat alle modellen FALSE retourneren hebben ze het dus ook in 99% van de tijd correct. Deze hoge nauwkeurigheid zegt dus helemaal niks.

9.5 Bijlage E - Data-analyse



Figuur 7 - Decisiontree modellen

2.3 Conclusie

Uit de analyses kan geconcludeerd worden dat het processor- en geheugen verbruik op momenten dat foutmeldingen zijn geregistreerd niet uitschieten ten opzichten van de andere verbruiken.

Bij het toepassen van classificatie algoritmes werd het duidelijk dat er geen fatsoenlijke modellen kunnen worden getraind aan de hand van de beschikbare dataset. Dit komt omdat er gebruik gemaakt moet worden van niet geclassificeerde dataset. De aannames die werden gedaan over wat afwijkend gedrag is, zijn waarschijnlijk niet correct.

Hieruit kan geconcludeerd worden dat het onmogelijk is aan de hand van classificatie algoritmes voorspellingen te maken voor de huidige dataset. Om wel voorspellingen te kunnen maken is er niet per definitie meer maar andere data noodzakelijk. Wanneer er geclassificeerde dataset beschikbaar is met daarin een (groot) aantal gemarkeerde fouten is het wellicht wel mogelijk door middel van classificatie algoritmes voorspellingen te maken.

3. Forecasting

Het classificeren van metingen was niet mogelijk. Daarom is er besloten te onderzoeken of het mogelijk is het (totale) processor en/of geheugen verbruik te voorspellen. De metingen van het (totale) processor verbruik bevatten de meeste fluctuaties. Er is daarom besloten als eerst te onderzoeken of het voorspellen van het (totale) processor verbruik mogelijk is.

In de statistische programmeertaal R zijn door middel van het "forecast" package het ARIMA, ETS en HoltWinters model beschikbaar *(Quick-R, z.d.)*. Dit zijn modellen die gebruikt kunnen worden voor het voorspellen van tijdreeks gegevens. Er is besloten niet verder te kijken naar andere modellen omdat er beperkte tijd beschikbaar is.

Het HoltWinters model bleek onbruikbaar omdat in dat model een seizoen gedefinieerd moet worden. In de andere twee modellen is dit optioneel. Door een seizoen te specifiëren kunnen bij bijvoorbeeld financiële cijfers het koopgedrag per seizoen meegenomen worden in de voorspelling. De metingen van het (totale) processor verbruik bevatten geen seizoenen.

3.1 ARIMA Model - ETS Model

ARIMA staat voor "Autoregressive integrated moving average" *(Hyndman, 2015)*. Het ARIMA model bestaat zoals de naam al zegt uit een autoregressive algoritme gecombineerd met een moving average algoritme (Nau, ARIMA models for time series forecasting, z.d.). Beide algoritmes voorspellen op hun eigen manier de toekomst en door deze te combineren zou er een "betere" voorspelling bereikt moeten worden.

ETS staat voor "Exponential smoothing state space model". Het grote verschil tussen ETS en ARIMA is dat bij ARIMA alle getallen het zelfde gewicht hebben. Bij ETS wegen minder recente metingen minder mee door een smoothing factor *(Nau, Moving average and exponential smoothing models, z.d.)*.

Beide modellen hebben een aantal verschillende varianten op hun model. Door middel van een functie in statistische programmeertaal R kan automatische de beste variant geselecteerd worden.

Uit de dataset is een handmatige subset geselecteerd met daarin zoveel mogelijk fluctuaties. Zo kan door middel van een experiment gekeken worden welk model de hoogste nauwkeurigheid heeft in een "moeilijke situatie". Om te bepalen welk model de hoogste nauwkeurigheid heeft zijn 1000 voorspellingen gemaakt over de subset. Voor alle voorspellingen samen is een gemiddelde nauwkeurigheid (per absolute afwijking) en gemiddelde absolute afwijking bepaald. Voor ieder model is de beste variant model gebruikt. De resultaten zijn te zien in

9.5 Bijlage E - Data-analyse

Tabel 1 - 1 step ahead (1000 voorspellingen), *Tabel 2 - 5 steps ahead (1000 voorspellingen)* en *Tabel 3 - 10 steps ahead (1000 voorspellingen)*. Het ETS model scoorde het beste.

9.5 Bijlage E - Data-analyse

Tabel 1 - 1 step ahead (1000 voorspellingen)

	% correct met absolute afwijking > 5		Gemiddelde absolute afwijking	
Training data	ARIMA	ETS	ARIMA	ETS
2 uur	47,60%	56,00%	8,71309	7,84458
3 uur	47,40%	55,50%	8,99049	7,95653
5 uur	44,50%	55,60%	9,32761	7,91266
8 uur	42,90%	56,00%	9,64481	7,85605

Tabel 2 - 5 steps ahead (1000 voorspellingen)

	% correct met absolute afwijking > 5		Gemiddelde absolute afwijking	
Training data	ARIMA	ETS	ARIMA	ETS
2 uur	36,08%	49,75%	13,10540	11,79164
3 uur	34,57%	49,65%	13,42563	11,76784
5 uur	31,46%	49,55%	14,03888	11,77718
8 uur	30,55%	49,55%	14,59830	11,78920

Tabel 3 - 10 steps ahead (1000 voorspellingen)

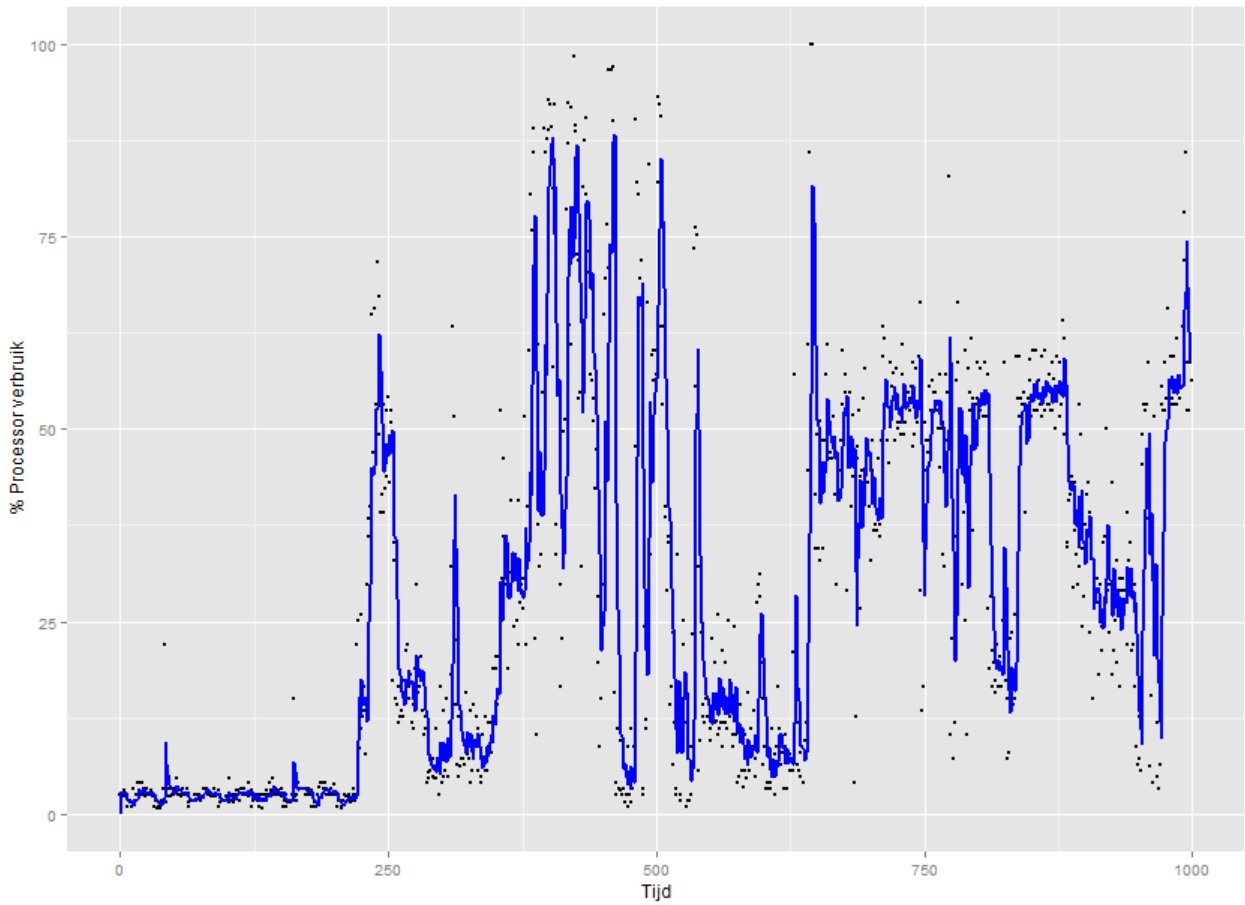
	% correct met absolute afwijking > 5		Gemiddelde absolute afwijking	
Training data	ARIMA	ETS	ARIMA	ETS
2 uur	33,94%	46,16%	14,09754	13,32810
3 uur	32,73%	45,86%	14,31935	13,24920
5 uur	29,90%	46,06%	14,82169	13,27896
8 uur	29,90%	46,16%	15,27710	13,31932

In *Tabel 4 - 1 step ahead (1000 voorspellingen) met 2 uur training data* zijn verdere details van de beste voorspelling (ETS model) te zien. De gemiddelde afwijking van de gemeten waardes ten opzichte van de voorspelde waardes is 7,84. Dat betekent dat een voorspelling 7,84 naar boven en na beneden af kan wijken. Wanneer een afwijking van 4 a 5 na boven en na beneden wordt gehanteerd kan maar 50 tot 56% correct voorspeld worden.

Tabel 4 - 1 step ahead (1000 voorspellingen) met 2 uur training data

Omschrijving	ETS
% correct met absolute afwijking > 1	23,70%
% correct met absolute afwijking > 2	34,90%
% correct met absolute afwijking > 3	44,90%
% correct met absolute afwijking > 4	50,60%
% correct met absolute afwijking > 5	56,00%
Gemiddelde absolute afwijking	7,84458

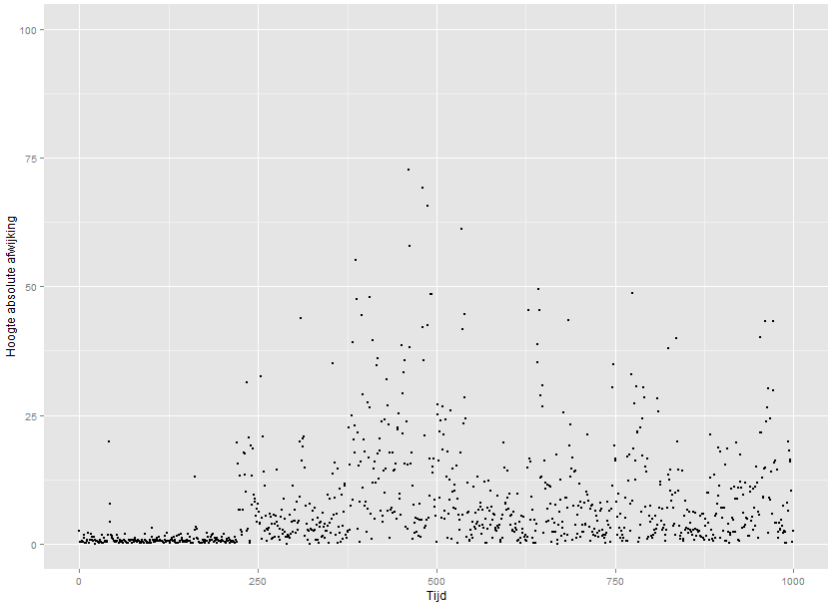
9.5 Bijlage E - Data-analyse



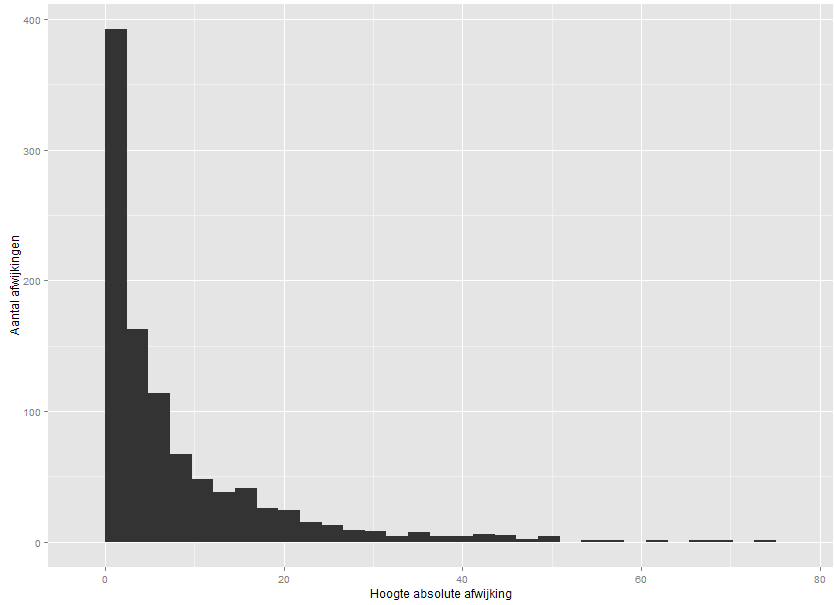
Figuur 8 - Voorspelling ETS model, 1 step ahead (1000 voorspellingen) met 2 uur training data

In *Figuur 8 - Voorspelling ETS model, 1 step ahead (1000 voorspellingen) met 2 uur training data* is de voorspelling van het ETS model (blauw lijn) en de gemeten waardes (zwarte punten) te zien. Op het oog ziet dit er redelijk goed uit maar als er naar de statistieken wordt gekeken valt dit tegen.

9.5 Bijlage E - Data-analyse



Figuur 9 - Absolute afwijking tussen gemeten en voorspelde waardes



Figuur 10 - Histogram absolute afwijkingen

In *Figuur 9 - Absolute afwijking tussen gemeten en voorspelde waardes* zijn de absolute afwijkingen te zien. Wat opvalt is dat de afwijkingen het grootst zijn bij de hoge pieken. Dat wil zeggen dat de pieken niet correct door het algoritme worden ontdekt. In *Figuur 10 - Histogram absolute afwijkingen* is een histogram te zien die de frequentie van alle absolute afwijkingen weergeeft. Hierin is een grote spreiding in de absolute afwijkingen te zien. Van alle voorspellingen is er maar ongeveer 40% dat minimaal (absolute afwijking onder de 1) afwijkt. Idealiter is de spreiding zo klein mogelijk met daarin zoveel mogelijk absolute afwijkingen aan de linkerkant (absolute afwijking onder de 1)van de grafiek.

3.2 ETS

Aangezien het niet bekend is of de dataset afwijkend gedrag bevat, kan het zijn dat de geselecteerde subset afwijkend gedrag vertoont en daarom de test beïnvloed. Om dit uit te sluiten zijn er 4 willekeurige subsets gekozen en zijn hierop 2.500 voorspellingen gemaakt. Voor het maken van voorspellingen is voor iedere subset 2 uur aan trainingsdata gebruikt. De resultaten hiervan zijn te zien in *Tabel 5 - Willekeurige subsets, 1 step ahead (2500 voorspellingen)*, *Tabel 6 - Willekeurige subsets, 5 steps ahead (2500 voorspellingen)*, *Tabel 7 - Willekeurige subsets, 10 steps ahead (2500 voorspellingen)* en *Tabel 8 - Willekeurige subsets, 30 steps ahead (2500 voorspellingen)*. Uit deze resultaten blijkt dat het relatief nauwkeurig voorspellen wel degelijk mogelijk is.

Tabel 5 - Willekeurige subsets, 1 step ahead (2500 voorspellingen)

Omschrijving	Subset 1	Subset 2	Subset 3	Subset 4	Gemiddeld
% correct met absolute afwijking > 1	80,96%	74,92%	76,04%	82,28%	78,55%
% correct met absolute afwijking > 2	91,96%	90,76%	88,72%	92,40%	90,96%
% correct met absolute afwijking > 3	94,56%	94,20%	91,92%	94,40%	93,77%
% correct met absolute afwijking > 4	95,32%	95,40%	93,32%	95,40%	94,86%
% correct met absolute afwijking > 5	95,80%	96,24%	94,64%	95,76%	95,61%
Gemiddelde absolute afwijking	1,19893	1,35351	1,51573	1,32033	1,34713

Tabel 6 - Willekeurige subsets, 5 steps ahead (2500 voorspellingen)

Omschrijving	Subset 1	Subset 2	Subset 3	Subset 4	Gemiddeld
% correct met absolute afwijking > 1	75,91%	66,33%	66,25%	77,07%	71,39%
% correct met absolute afwijking > 2	89,98%	88,86%	83,93%	90,46%	88,31%
% correct met absolute afwijking > 3	93,27%	93,15%	88,74%	93,27%	92,10%
% correct met absolute afwijking > 4	94,11%	94,39%	90,86%	94,07%	93,36%
% correct met absolute afwijking > 5	94,63%	95,31%	92,18%	94,75%	94,22%
Gemiddelde absolute afwijking	1,53107	1,69367	2,23022	1,87937	1,83358

Tabel 7 - Willekeurige subsets, 10 steps ahead (2500 voorspellingen)

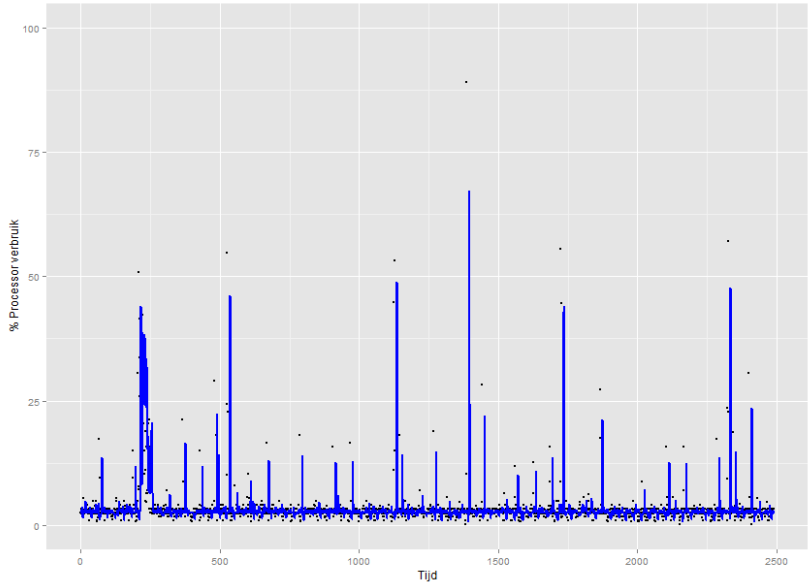
Omschrijving	Subset 1	Subset 2	Subset 3	Subset 4	Gemiddeld
% correct met absolute afwijking > 1	74,78%	56,67%	55,22%	73,57%	65,06%
% correct met absolute afwijking > 2	88,96%	85,58%	79,48%	88,80%	85,70%
% correct met absolute afwijking > 3	92,57%	93,05%	86,18%	92,05%	90,96%
% correct met absolute afwijking > 4	93,86%	94,46%	89,28%	93,65%	92,81%
% correct met absolute afwijking > 5	94,42%	95,26%	91,61%	94,14%	93,86%
Gemiddelde absolute afwijking	1,65407	1,81991	2,61549	2,12004	2,05238

9.5 Bijlage E - Data-analyse

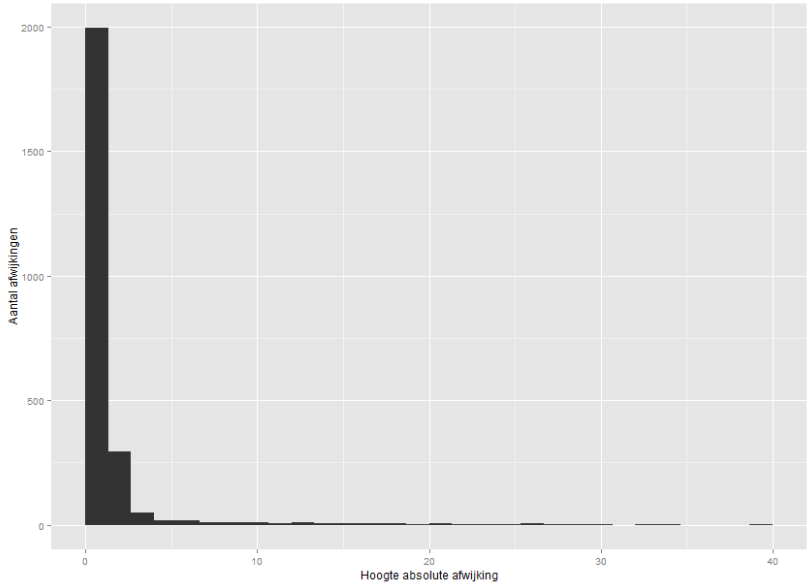
Tabel 8 - Willekeurige subsets, 30 steps ahead (2500 voorspellingen)

Omschrijving	Subset 1	Subset 2	Subset 3	Subset 4	Gemiddeld
% correct met absolute afwijking > 1	78,46%	70,20%	35,22%	73,28%	64,29%
% correct met absolute afwijking > 2	89,07%	90,08%	59,76%	87,69%	81,65%
% correct met absolute afwijking > 3	92,02%	93,60%	72,47%	91,30%	87,35%
% correct met absolute afwijking > 4	93,00%	94,33%	79,43%	93,00%	89,94%
% correct met absolute afwijking > 5	93,60%	95,14%	84,13%	94,05%	91,73%
Gemiddelde absolute afwijking	1,83303	1,62851	4,43082	2,21238	2,52619

9.5 Bijlage E - Data-analyse



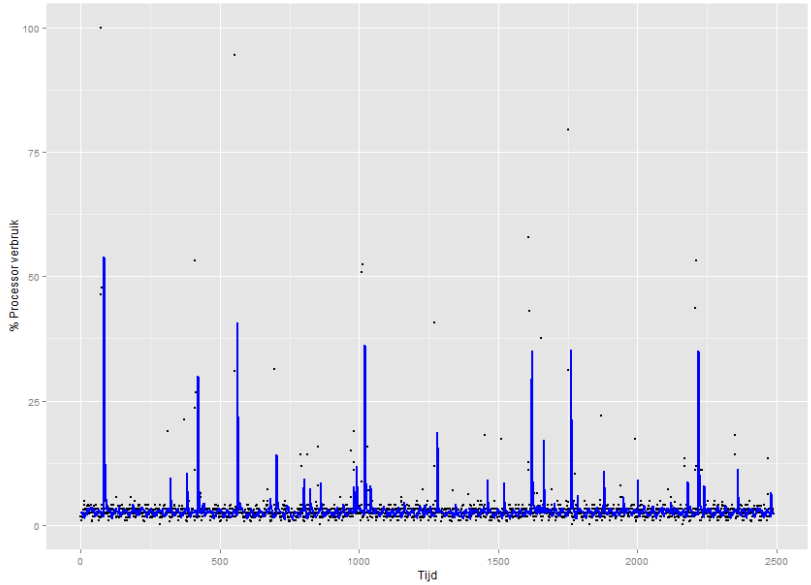
Figuur 11 - Voorspelling subset 1, 10 steps ahead (2000 voorspellingen)



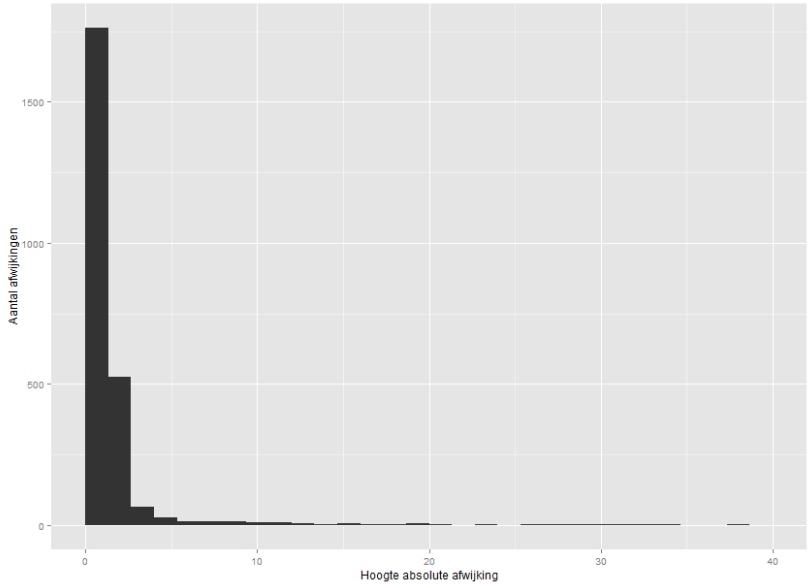
Figuur 12 - Histogram absolute afwijkingen subset 1

In *Figuur 11 - Voorspelling subset 1, 10 steps ahead (2000 voorspellingen)* is subset 1 te zien met daarin de voorspelling (blauw lijn) en de gemeten waardes (zwarte punten). In *Figuur 12 - Histogram absolute afwijkingen subset 1* is een histogram te zien die frequentie van alle absolute afwijkingen voor de voorspelling over subset 1 weergeeft.

9.5 Bijlage E - Data-analyse



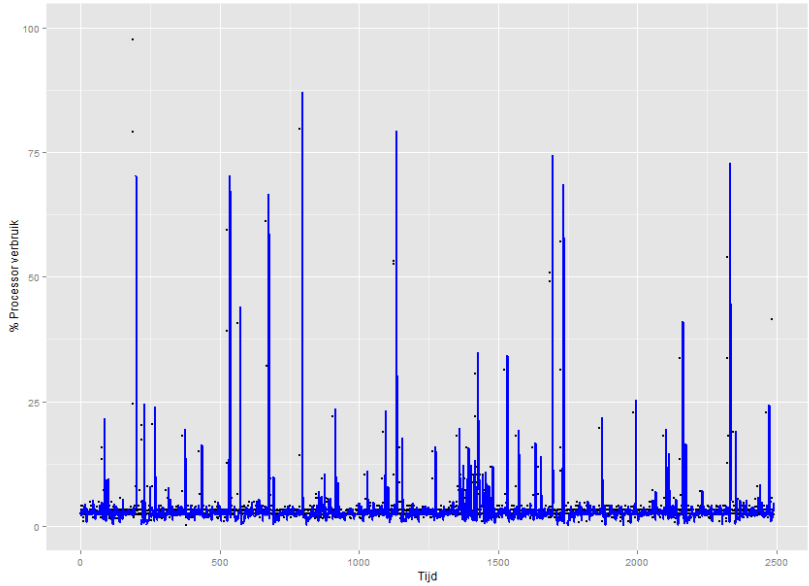
Figuur 13 - Voorspelling subset 2, 10 steps ahead (2000 voorspellingen)



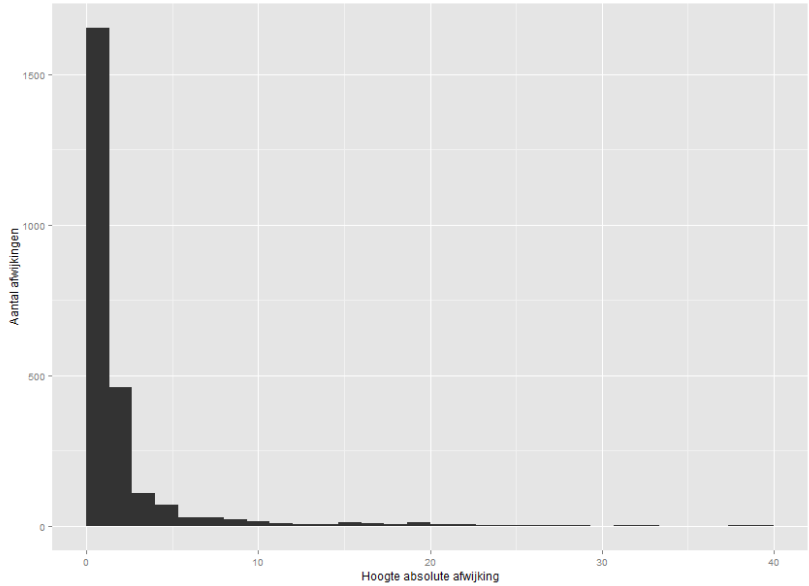
Figuur 14 - Histogram absolute afwijkingen subset 2

In *Figuur 13 - Voorspelling subset 2, 10 steps ahead (2000 voorspellingen)* is subset 2 te zien met daarin de voorspelling (blauw lijn) en de gemeten waardes (zwarte punten). In *Figuur 14 - Histogram absolute afwijkingen subset 2* is een histogram te zien die frequentie van alle absolute afwijkingen voor de voorspelling over subset 2 weergeeft.

9.5 Bijlage E - Data-analyse



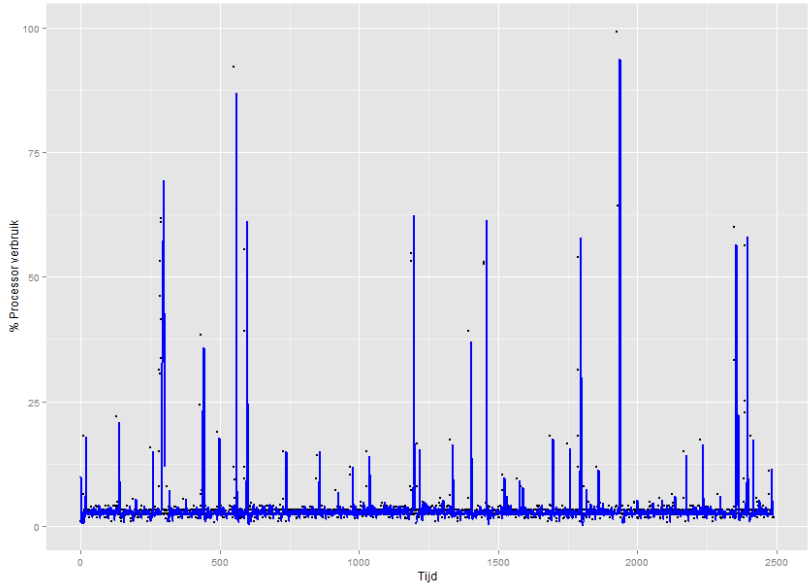
Figuur 15 - Voorspelling subset 3, 10 steps ahead (2000 voorspellingen)



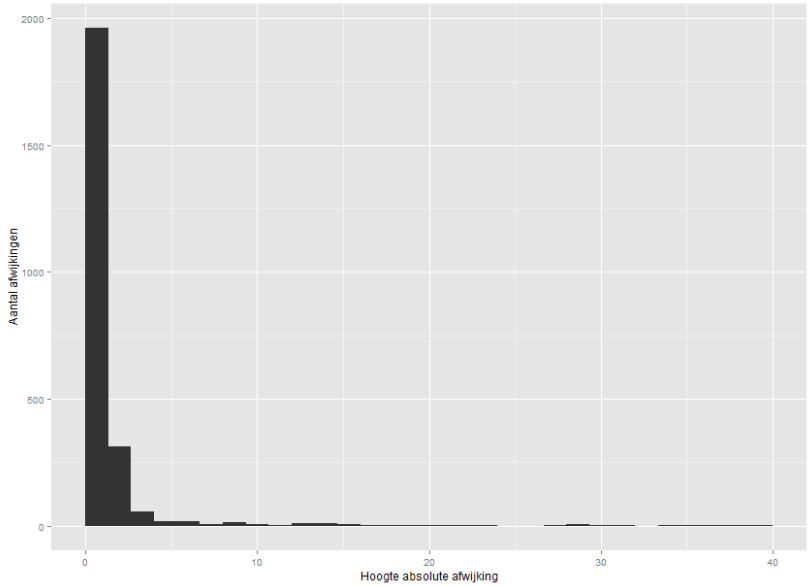
Figuur 16 - Histogram absolute afwijkingen subset 3

In *Figuur 15 - Voorspelling subset 3, 10 steps ahead (2000 voorspellingen)*Figuur 11 - Voorspelling subset 1, 10 steps ahead (2000 voorspellingen) is subset 3 te zien met daarin de voorspelling (blauw lijn) en de gemeten waardes (zwarte punten). In *Figuur 12 - Histogram absolute afwijkingen subset 1* is een histogram te zien die frequentie van alle absolute afwijkingen voor de voorspelling over subset 3 weergeeft.

9.5 Bijlage E - Data-analyse



Figuur 17 - Voorspelling subset 4, 10 steps ahead (2000 voorspellingen)

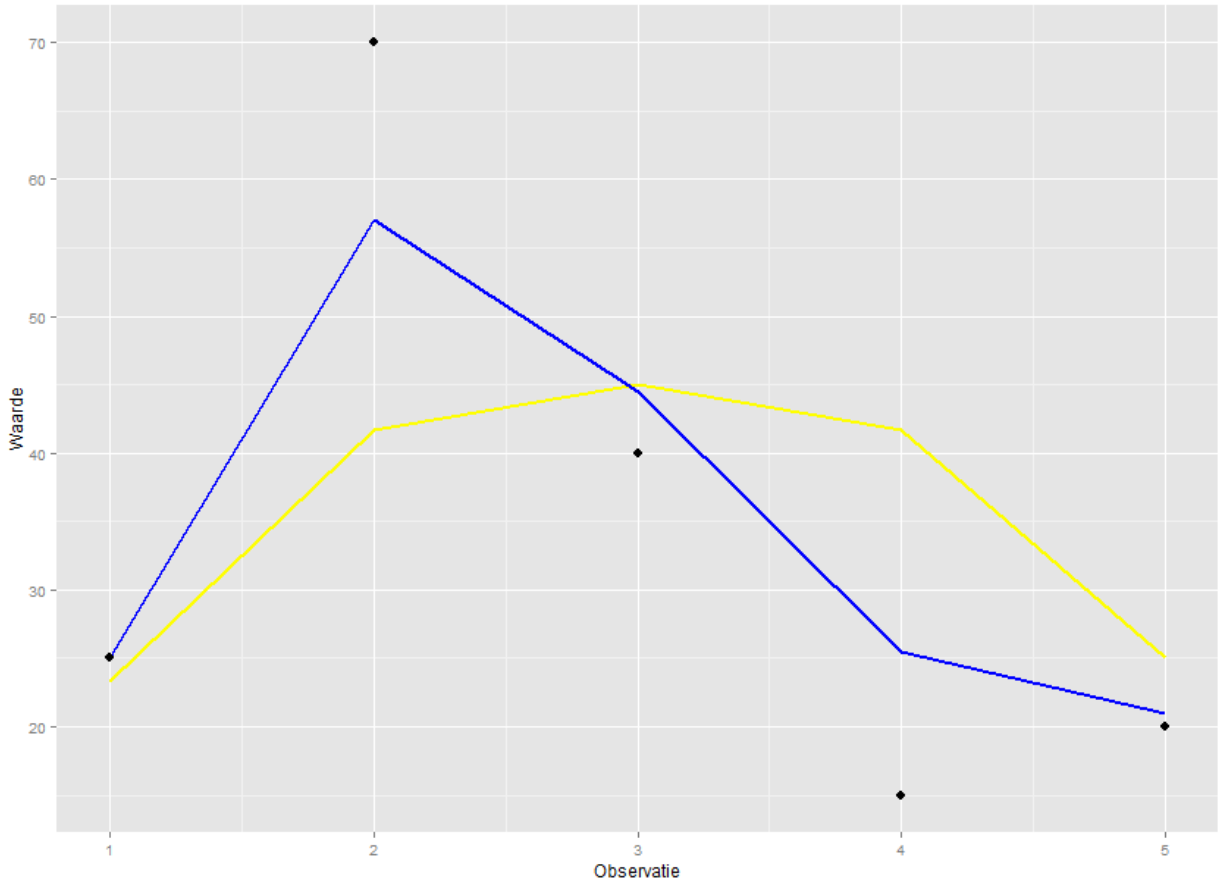


Figuur 18 - Histogram absolute afwijkingen subset 4

In *Figuur 17 - Voorspelling subset 4, 10 steps ahead (2000 voorspellingen)* is subset 4 te zien met daarin de voorspelling (blauw lijn) en de gemeten waardes (zwarte punten). In *Figuur 18 - Histogram absolute afwijkingen subset 4* is een histogram te zien die frequentie van alle absolute afwijkingen voor de voorspelling over subset 4 weergeeft.

3.3 Conclusie

Het voorspellen van het (totale) processor verbruik is daarin tegen mogelijk. De nauwkeurigheid van de voorspellingen is echter moeilijk te bepalen aangezien het niet bekend is of de dataset afwijkend gedrag bevat. Bij het bepalen van welk model het meest geschikt is voor het voorspellen van het (totale) processor verbruik werd het duidelijk dat het ETS model betere resultaten biedt dan het ARIMA model. Dit komt waarschijnlijk door het feit dat het (totale) processor verbruik veel en onregelmatig fluctueert. Het ETS model doet in tegenstelling tot het ARIMA model gewichten hangen aan gemeten waardes. Omdat nieuwere waardes zwaarder meetellen dan oudere waardes kunnen fluctuaties beter voorspeld worden.



Figuur 19 - Verschil Simple Moving Average ten opzichte van Exponential Smoothing

In *Figuur 19 - Verschil Simple Moving Average ten opzichte van Exponential Smoothing* is een simpel voorbeeld te zien waarbij het verleden bestaat uit twee observaties: 15 gevolgd door 30. De gele lijn weergeeft een Simple Moving Average die 3 observaties terug kijkt. De blauwe lijn weergeeft een Exponential Smoothing waarbij de meest recente waarde voor 70% mee weegt, de observatie daar voor 20% en de observatie daar voor 10%.

4. Conclusie en aanbevelingen

Uit analyses en pogingen tot het classificeren van de dataset werd al snel duidelijk dat de dataset waarschijnlijk onbruikbaar is. De modellen die getraind werden op de geclassificeerde datasets waren onbruikbaar. De aannames die werden gedaan over wat afwijkend gedrag is, zijn waarschijnlijk niet correct. Het toepassen van classificatie algoritmes is alleen mogelijk wanneer er een geclassificeerde dataset beschikbaar is. Het voorspellen van afwijkend gedrag is daarom niet mogelijk.

Het voorspellen van het (totale) processor verbruik is daarin tegen wel mogelijk. De nauwkeurigheid van de voorspellingen is echter moeilijk te bepalen aangezien het niet bekend is of de dataset afwijkend gedrag bevat. Ondanks er een lage nauwkeurigheid en hoge gemiddelde absolute afwijking was geconstateerd voor een handmatig geselecteerde subset gaven de willekeurige subsets hoge nauwkeurigheden en lage gemiddelde absolute afwijkingen. Het is dus mogelijk dat de handmatig geselecteerde subset toevallig afwijkend gedrag bevat en daarom resulteert in een lagere nauwkeurigheid en hogere gemiddelde absolute afwijking.

Ondanks het voorspellen van het (totale) processor verbruik mogelijk lijkt te zijn gaat het advies echter eerst na het verkrijgen en analyseren van een geclassificeerde dataset. Uit de analyses op een geclassificeerde dataset moet het duidelijk worden hoe nauwkeurig het voorspellen van het (totale) processor verbruik daadwerkelijk is. De huidige dataset kan afwijkend gedrag vertonen wat ervoor zorgt dat de nauwkeurigheid dan de voorspelling omlaag gaat. Wanneer afwijkend gedrag uitgesloten wordt kan de echte nauwkeurigheid bepaald worden.

Het voorspellen van het (totale) processor verbruik kan resulteren in het voorspellen van afwijkend gedrag dat ontstaat door hoge of afwijkende verbruiken. Er kan namelijk worden voorspeld dat er voor een x periode een y verbruik gaat plaatsvinden. Of en hoe afwijkend gedrag gerelateerd is aan het (totale) processor verbruik of andere verbruiken is echter nog onbekend.

Door classificatie algoritmes toe te passen op een geclassificeerde dataset moet het duidelijk worden of er bruikbare modellen getraind kunnen worden en hoe gegevens gerelateerd zijn aan elkaar. Als er bruikbare modellen uitkomen wordt het voorspellen van afwijkend gedrag mogelijk.

5. Literatuurlijst

Hyndman, R. (2015, oktober 20). *Forecasting Functions for Time Series and Linear Models.* Opgeroepen op december 14, 2015, van R-Project: https://cran.r-project.org/web/packages/forecast/forecast.pdf

Nau, R. (z.d.). *ARIMA models for time series forecasting.* Opgeroepen op december 14, 2015, van Fuqua School of Business, Duke University: http://people.duke.edu/~rnau/411arim.htm

Nau, R. (z.d.). *Moving average and exponential smoothing models.* Opgeroepen op december 14, 2015, van Fuqua School of Business, Duke University: http://people.duke.edu/~rnau/411avg.htm

Quick-R. (z.d.). *Time Series and Forecasting.* Opgeroepen op december 16, 2015, van Quick-R: http://www.statmethods.net/advstats/timeseries.html

Quick-R. (z.d.). *Tree-Based Models.* Opgeroepen op december 14, 2015, van Quick-R: http://www.statmethods.net/advstats/cart.html

Wesley. (2013, april 29). *A Brief Tour of the Trees and Forests.* Opgeroepen op december 14, 2015, van R-bloggers: http://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/

9.6 Bijlage F - Specs ontwikkelomgeving

Omschrijving	
Besturingssysteem	Windows 8.1 Enterprise
Processor	Intel Core 2 Duo E6850 (3.0GHZ)
Geheugen	8 GB DDR2 (400 MHz)
Opslag	RAID 0 configuratie van twee 5400RPM SATA-2 harde schijven

Integratie van de statistische programmeertaal R met .NET

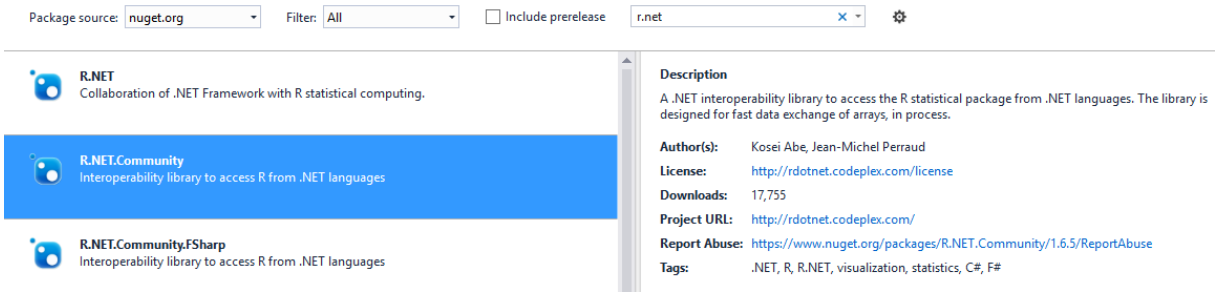
De term "Machine Learning" hoor je steeds vaker. Wanneer je hiermee zelf aan de slag wilt gaan, kom je al snel een oplossing als Azure Machine Learning tegen. Misschien volg je een online cursus en ga je aan de slag met Python of de statistische programmeertaal R. Allemaal leuk en aardig maar wat als je nu iets “cools” gevonden hebt en dit wilt integreren in bijvoorbeeld een bestaand Microsoft applicatie landschap? Ja het is via een paar klikken mogelijk om een model gemaakt in Azure Machine Learning te publiceren als REST service. Maar wat als je niet in de Cloud wilt werken of net wat meer wilt?

In dit artikel wil ik aan de hand van een voorbeeld laten zien hoe je een (.NET) applicatie ontwikkelt die koerswaardes kan voorspellen. Het model, getraind in de statistische programmeertaal R, waarmee voorspellingen worden gemaakt moet automatisch bijgewerkt worden wanneer nieuwe koerswaardes bekend zijn.

Benodigdheden

Om zelf van start te kunnen gaan heb je het volgende nodig:

- R moet geïnstalleerd zijn op de machine waarop de applicatie uitgevoerd gaat worden. Heb je R nog niet geïnstalleerd, dan kun je het hier downloaden: https://cran.rstudio.com/.
- Het R.NET.Community framework moet toegevoegd worden aan het (.NET) project. Deze is te verkrijgen via de NuGet package manager (zie *Figuur 1 - R.NET.Community*).



Figuur 1 - R.NET.Community

- Tot slot heb je data nodig. Er zijn online een hoop koerswaardes te vinden. Wanneer je snel aan de slag wilt kan je de koerswaardes van Yahoo gebruiken. Yahoo stelt haar koerswaardes van de afgelopen jaren als CSV beschikbaar op: http://finance.yahoo.com/q/hp?s=YHOO.

Over R.NET.Community

Dit framework bestaat al sinds december 2010. Om de integratie van R met .NET mogelijk te maken wordt er gebruik gemaakt van de geïnstalleerde R library (R.dll).

Hoewel dit framework goed lijkt te werken heb ik een tweetal beperkingen gevonden:

- Het is niet mogelijk meerdere zogeheten REngine instanties aan te maken. Hier kun je omheen werken door gebruik te maken van meerdere environments in R zodat je in feite meerdere gescheiden scopes introduceert binnen die enkele REngine.
- Het is niet mogelijk een REngine instantie te herstarten.

Meer informatie over environments in R kan je vinden op: http://adv-r.had.co.nz/Environments.html.

9.7 Bijlage G - Artikel KnowNow

Programmacode

Het uitvoeren van R programmacode vanuit .NET doe je met behulp van de REngine. Voordat je deze REngine instantie kan aanroepen, moet het pad naar de R.dll en R home directory ingesteld zijn. Hieronder is te zien hoe dit in .NET uitziet.

```
REngine.SetEnvironmentVariables(); // Bij standaard instellingen
REngine.SetEnvironmentVariables(@"X:\R\R-3.2.2\bin\i386\"); // Bij aangepaste instellingen
REngine engine = REngine.GetInstance();
```

Vervolgens kan via de Evaluate methode van de REngine instantie, R programmacode uitgevoerd worden. De methode retourneert vervolgens het resultaat van de uitgevoerde R programmacode. In *Tabel 1 - Vertaling datatypes R van en naar .NET* is de vertaling tussen datatypes weergeven (bron: http://rdotnet.codeplex.com/documentation).

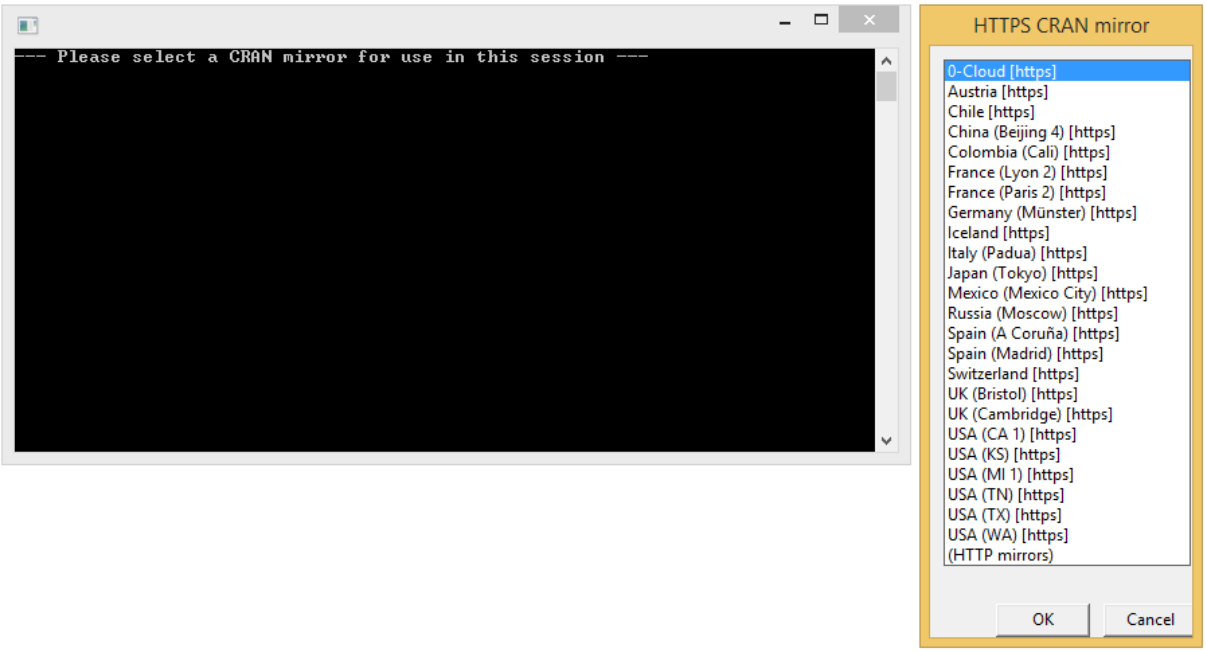
R	R.NET	.NET Framework
character vector	RDotNet.CharacterVector	System.String[]
integer vector	RDotNet.IntegerVector	System.Int32[]
real vector	RDotNet.NumericVector	System.Double[]
complex vector	RDotNet.ComplexVector	System.Numerics.Complex[]
raw vector	RDotNet.RawVector	System.Byte[]
logical vector	RDotNet.LogicalVector	System.Boolean[]
character matrix	RDotNet.CharacterMatrix	System.String[,]
integer matrix	RDotNet.IntegerMatrix	System.Int32[,]
real matrix	RDotNet.NumericMatrix	System.Double[,]
complex matrix	RDotNet.ComplexMatrix	System.Numerics.Complex[,]
raw matrix	RDotNet.RawMatrix	System.Byte[,]
logical matrix	RDotNet.LogicalMatrix	System.Boolean[,]
list	RDotNet.GenericVector	
data frame	RDotNet.GenericVector	
data frame	RDotNet.DataFrame	
function	RDotNet.Function	
factor	RDotNet.Factor	System.Int32[]
S4	RDotNet.S4Object	

Tabel 1 - Vertaling datatypes R van en naar .NET

Bij het uitvoeren van R programmacode kan het zijn dat er een dialoogvenster verschijnt. Dit gebeurt bijvoorbeeld bij het installeren van een package (zie *Figuur 2 - Dialoogvenster*).

```
engine.Evaluate("install.packages(\"forecast\")");
engine.Evaluate("library(forecast)");
```

9.7 Bijlage G - Artikel KnowNow



Figuur 2 - Dialoogvenster

Dit is op te lossen door expliciet te specifiëren vanaf welke repository de package gedownload moet worden. Daarnaast hoeft een package maar één keer geïnstalleerd te worden.

```
engine.Evaluate("if (!require(\"forecast\")) install.packages(\"forecast\", repos =
\"http://cran.uk.r-project.org\")");
```

Tot slot moet je niet vergeten de REngine op te ruimen. Hiervoor kan je een using block gebruiken, maar na het opruimen van de REngine instantie is het niet meer mogelijk een nieuwe instantie te starten.

```
engine.Dispose();
```

Voor meer informatie, zie documentatie van R.NET.Community op : http://jmp75.github.io/rdotnet/.

9.7 Bijlage G - Artikel KnowNow

Een volledig voorbeeld.

```
public class StockPricePredictor
{
    /// <summary>
    /// The REngine instance
    /// </summary>
    private REngine engine;

    /// <summary>
    /// Initialize a new StockPricePredictor instance
    /// </summary>
    public StockPricePredictor()
    {
        REngine.SetEnvironmentVariables();
        this.engine = REngine.GetInstance();
        this.engine.AutoPrint = false;
        this.engine.Evaluate("if (!require(\"forecast\")) install.packages(\"forecast\",
                              repos = \"http://cran.uk.r-project.org\")");
        this.engine.Evaluate("library(forecast)");
    }

    /// <summary>
    /// Train the model by providing stock prices from the past
    /// </summary>
    /// <param name="stockPrices">The stock prices from the past</param>
    public void TrainModel(IEnumerable<double> stockPrices)
    {
        this.engine.SetSymbol("stockPrices", this.engine.CreateNumericVector(stockPrices));
        this.engine.Evaluate("model = auto.arima(stockPrices)");
    }

    /// <summary>
    /// Predict future stock prices
    /// </summary>
    /// <returns></returns>
    public double[] Predict()
    {
        GenericVector prediction = this.engine.Evaluate("predict(model, 3)").AsList();
        return this.prediction["pred"].AsNumeric().ToArray();
    }

    /// <summary>
    /// Update the model by providing new stock prices
    /// </summary>
    /// <param name="newStockPrices"></param>
    public void UpdateModel(IEnumerable<double> newStockPrices)
    {
        this.engine.SetSymbol("newStockPrices", this.engine.CreateNumericVector(newStockPrices));
        this.engine.Evaluate("stockPrices = c(stockPrices, newStockPrices)");
        this.engine.Evaluate("model = Arima(stockPrices, model=model)");
    }

    /// <summary>
    /// Remove the StockPricePredictor instance
    /// </summary>
    ~StockPricePredictor()
    {
        this.engine.Dispose();
    }
}
```