Software Architecture Document

Djimaro Talahatu

Djimaro Talahatu - 563631
Hogeschool van Arnhem en Nijmegen, faculteit ICA, HBO-ICT Software Development
Docentbegeleider: Dennis Breuker
Bedrijfsbegeleider: Alex Bijsterveld
Opdrachtgever: Cito
3 mei 2020, Apeldoorn
Versie 1.0

Inhoudsopgave

1 Definitions of Terms and Acronyms

Term	Definition
Onderzoeker	Het personeel van Cito dat helpt bij het ontwikkelen van toetsen en examens.
Niveau	Een categorie voor een verzameling variabelen. Bijvoorbeeld: De categorie 'school' heeft de variabelen HAN en Radboud. De categorie 'response' heeft antwoorden van leerlingen als variabelen.
Variabele	Een element met informatie. Bijvoorbeeld: "15", "Apeldoorn", "HBO" of "De hoofdstad van Amerika is Washington D.C.".
Eigenschap	De eigenschappen van variabelen. Bij een antwoord (response) van een leerling kan dit de lengte van de tekst zijn. Voor een set van voorbeelden zie (Zie Bijlage D)
Bestand	Een bestand met kolommen en rijen. Dit kan een Excel bestand zijn. In de eerste rij van een bestand staan de niveaus. Elke rij daaronder bevat variabelen.
Script	Een script dat de eigenschappen van variabelen achterhaalt en omzet naar een plaatje.
Open vraag	Een vraag waarop een leerling op zijn manier een antwoord (response) kan geven.
MoSCoW	Methode om requirements te prioriteren. Hierin staat de M voor Must haves, de S voor Should haves, de C voor Could haves, en de W voor Would haves.
Response	Is hetzelfde als een antwoord van een leerling.
Variable	Is hetzelfde als variabele.
Levels	Is hetzelfde als niveaus.
Property	Is hetzelfde als eigenschap.

Tabel 1.1: Definities

2 Inleiding

Cito maakt het onderwijs krachtiger door te laten zien wat leerlingen in hun mars hebben. Ze zetten hun unieke expertise in om de ontwikkeling van kennis, vaardigheden en competenties te volgen en inzichtelijk te maken. Daarmee kunnen zij leraren, ouders en werkgevers ondersteunen. Op deze manier kunnen leerlingen het beste uit zichzelf halen en richting geven aan hun toekomst.

Cito laat zien wat leerlingen in hun mars hebben door het ontwikkelen van toetsen en examens en de resultaten vervolgens te meten met hulpmiddelen zoals meetinstrumenten. Om te laten zien wat leerlingen kunnen, moet de kwaliteit van toetsen en examens goed zijn. De kwaliteit van toetsen en examens onder ander wordt bepaald door onder ander de antwoorden die leerlingen geven op een onderdeel van een toets of examen.

Het doel van Cito is het ervoor zorgen dat iedere leerling gelijke kansen krijgt. Dit is, sinds de oprichting, de drijfveer voor medewerkers van Cito. De primaire focus van 'goed en eerlijk toetsen' is verbreed naar het objectief meten van mogelijkheden en talenten.

Toetsen en examens bestaan uit open en gesloten vragen. Bij een open vraag kan de leerling op zijn manier antwoord geven. Bij een gesloten vraag kan de leerling kiezen uit een set van ant-

woorden.

Er zijn meetinstrumenten die de kwaliteit van gesloten vragen kunnen meten. Bij een gesloten vraag is er een beperkte set van antwoorden beschikbaar. Deze set van antwoorden ondersteunen de interpretatie van de vraag. Hierdoor is het gemakkelijk om de antwoorden van leerlingen te analyseren met behulp van meetinstrumenten.

Het is moeilijker om de kwaliteit van open vragen te beoordelen. Dit geldt nog meer voor Nederlandse open vragen. Hiervoor zijn er nog minder hulpmiddelen beschikbaar.

De kwaliteit van Nederlandse open vragen worden op dit moment minder goed gewaarborgd, omdat het niet duidelijk is op welke eigenschappen er gelet moet worden. Voor dit project heeft Cito een open vraag(Zie Bijlage A) met daarbij antwoorden(Zie Bijlage B) gegeven. Deze open vraag gaat over het maken van een samenvatting in minder dan 150 woorden. Om te bepalen welke eigenschappen er nodig zijn om de kwaliteit van de open vraag te waarborgen is er onderzoek gedaan.(Zie Bijlage C) Hiervoor is een taalcoach van de HAN geïnterviewd.(Zie Bijlage D) Uit het interview bleek dat er geen theorie is over welke eigenschappen er nodig zijn. Wel is er advies gegeven om te kijken naar een set van eigenschappen. (Zie Bijlage D)

De kwaliteit van Nederlandse open vragen worden op dit moment minder goed gewaarborgd, omdat het meten van de kwaliteit zonder meetinstrumenten teveel tijd kost. Iemand moet een grote hoeveelheid antwoorden verzamelen en deze antwoorden en de eigenschappen ervan analyseren. Dit handmatig proces moet iedere keer gebeuren als er een nieuwe vraag is of als de vraag wordt gewijzigd.

Als de kwaliteit minder goed wordt gewaarborgd, kunnen de vragen op verschillende manieren worden geïnterpreteerd. Bij een open vraag kan de leerling op zijn manier antwoord geven. In vergelijking met gesloten vragen zijn er minder hulpmiddelen voor de leerling beschikbaar. Hierdoor is het lastiger voor de leerling om de vraag op de juiste manier te interpreteren.

Als de vragen op verschillende manieren kunnen worden geïnterpreteerd, hebben de leerlingen dus geen gelijke kansen. Elke leerling is anders en begrijpt de vraag op zijn eigen manier. Hierdoor kunnen leerlingen verkeerde antwoorden geven en misschien zelfs een toets niet halen. Dit beïnvloedt de toekomst van de leerling. Als de leerlingen geen gelijke kansen hebben is dit dus in strijd met het doel van Cito.

Het probleem is dus dat de kwaliteit van Nederlandse open vragen minder goed wordt gewaarborgd, omdat het proces teveel tijd kost en er geen meetinstrument voor beschikbaar is. De opdrachtgever van dit project is Eva de Schipper. Zij is een onderzoeker van Cito. Dit document is opgesteld aan de hand van gesprekken met de opdrachtgever. De uitvoerder van dit project is Djimaro Talahatu. Hij is een afstudeerder van de HAN. Dit document beschrijft de architectuur van de applicatie dat bijdraagt aan het oplossen van het probleem.

3 System Overview

De kwaliteit van een open vraag wordt bepaald door het analyseren van de antwoorden van leerlingen. Bij bijvoorbeeld de vraag "Wat is de hoofdstad van China?"wordt er Beijing of Peking als antwoord verwacht. Als het grootste deel van de leerlingen een totaal ander antwoord geven dan is er waarschijnlijk iets mis met de vraag.

Om te achterhalen waar er op gelet moet worden bij deze antwoorden is er een expert op het gebied van schrijven geïnterviewd. Hiervoor is Wout Waanders ,een taalcoach van de HAN, geïnterviewd. (Zie Bijlage D)

De taalcoach geeft aan dat er geen theorie is die bevestigt dat deze teksteigenschappen de enige of de juiste teksteigenschappen zijn, omdat iedereen op zijn eigen manier schrijft. Hij geeft aan dat een goede tekst kan bestaan uit veel of weinig woorden met korte of lange zinnen. Om deze reden is het lastig om te bepalen of een tekst goed is. Hij raadt aan dat voor het bepalen van de kwaliteit van een open vraag het handig is om de teksteigenschappen (Zie Bijlage D) van antwoorden te analyseren.

Als het niet duidelijk is welke teksteigenschappen de kwaliteit van open vragen inzichtelijk maken, moet er nog onderzoek gedaan worden. De aanbevolen teksteigenschappen zijn niet onderbouwd met onderzoeken. Hierom moet er één of moeten er meerdere onderzoeken gedaan worden om te bepalen wat een antwoord goed maakt en welke teksteigenschappen dat bepalen.

Het doen van onderzoek naar teksteigenschappen is niet haalbaar of relevant voor dit afstudeerproject. Hierom kan de applicatie worden uitgebreid met andere teksteigenschappen. Het onderzoeken van alle teksteigenschappen valt buiten het project, omdat het niet bijdraagt aan het halen van dit afstudeerproject. Doordat de onderzoekers eigenschappen kunnen toevoegen, wijzigen en verwijderen zijn ze niet beperkt tot de teksteigenschappen die tijdens dit project zijn achterhaald.

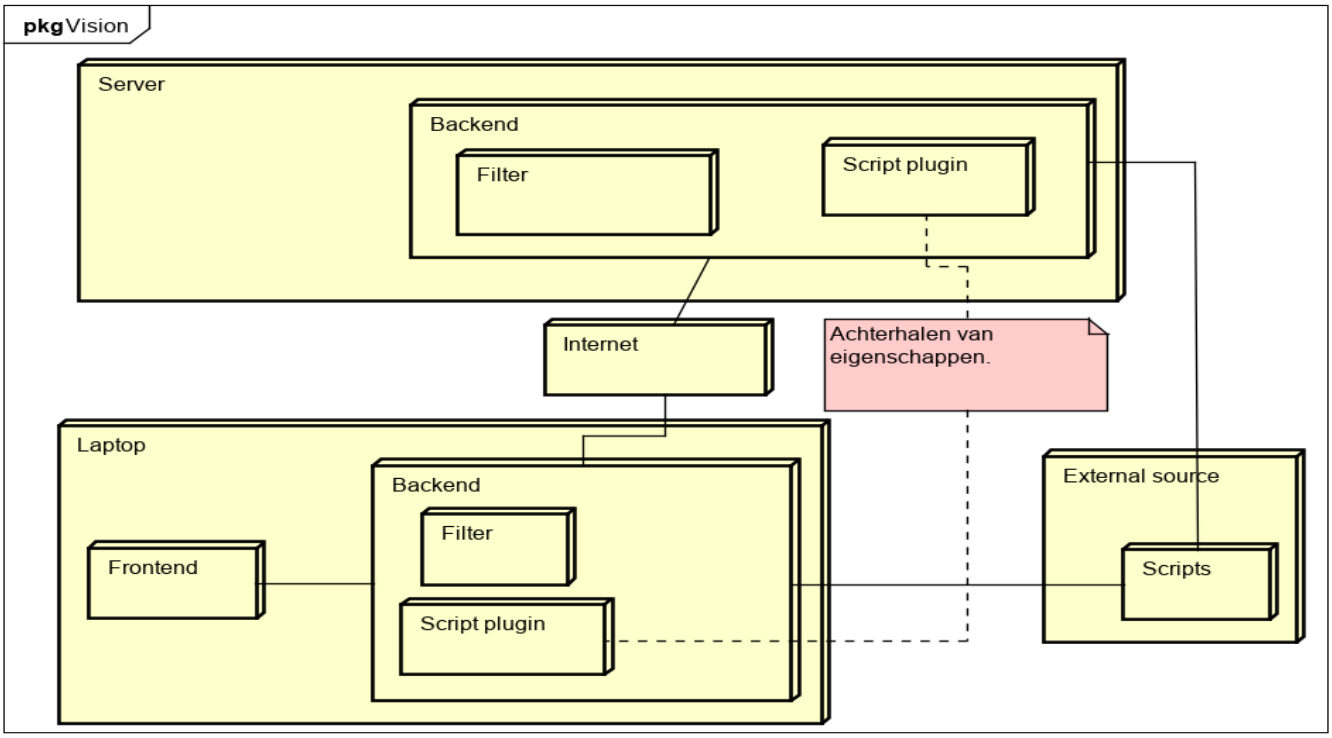
Door de eigenschappen te visualiseren kunnen de onderzoekers op basis van een grote hoeveelheid antwoorden de kwaliteit bepalen. Doordat de applicatie uitgebreid kan worden en de teksteigenschappen gevisualiseerd worden, worden er twee processen versnelt: het onderzoeken welke teksteigenschappen er nodig zijn en het proces om de kwaliteit te bepalen.

De applicatie uitbreiden met andere teksteigenschappen en het visualiseren ervan zijn één van de factoren die de structuur van de applicatie bepalen. Er zijn een aantal factoren die de structuur van de applicatie bepalen. In dit hoofdstuk worden alleen de drivers behandeld. De rest van deze factoren zijn onderdeel van hoofdstuk 6 op pagina 16.

Drivers zijn één van de factoren die invloed hebben op de structuur van de applicatie. Deze drivers zijn verkregen door de requirements, zie hiervoor hoofdstuk 5.2 op pagina 9 en concerns 4 op pagina 6 te analyseren. Hieronder staan de drivers die de structuur van de applicatie bepalen.

Scripts	De onderzoekers willen bij het gebruik van deze applicatie hun eigen scripts gebruiken. Hierbij willen niet alle onderzoekers een nieuwe taal leren. Hier kunnen zij vanwege de plugin architectuur de applicatie uitbreiden met nieuwe programmeertalen.
Repository	De Onderzoekers wisselen scripts onderling uit. Hierbij is het probleem dat het script niet altijd werkt. Om dit te garanderen worden de scripts centraal opgeslagen.
Bestand	De onderzoekers zijn gewend om in een bestand te werken met kolommen en tabellen. Door de plugin architectuur kunnen de onderzoekers de applicatie uitbreiden met andere soorten bestanden waarin zij werken met kolommen en rijen. Hierdoor is de applicatie meer toekomstbestendig.
Filteren	De onderzoekers willen niet telkens een nieuw bestand inladen. Hierom zou het gemakklijk zijn om een volledige bestand in te laden en dat het daarna te filteren.

Door de bovenstaande drivers ziet de applicatie er op architectuur niveau zo uit:



Figuur 3.1: Initial vision

Frontend	Graphical user interface van de software.
Filter	Is verantwoordelijk voor het inladen en filteren van een bestand.
Script plugin	Is verantwoordelijk voor het uitvoeren van het script, installeren van de benodigde libraries waar de script wordt uitgevoerd en het updaten van scripts.
Repository	Is verantwoordelijk voor het opslaan van de scripts.
Laptop	Laptop van een gebruker. Is verantwoordelijk voor het uitvoeren van een deel van de applicatie.
Server	Is verantwoordelijk voor het uitvoeren van scripts als de laptop het niet aankan.

4 Main Stakeholders and Concerns

Stakeholders zijn projectleden die een aandeel hebben op het project. De stakeholders van dit project zijn Djimaro, Eva en de onderzoekers van Cito.

Elke stakeholder heeft zorgen bij dit project. Deze zorgen zijn aantoonbaar meegenomen tijdens het ontwerpen van deze applicatie.

Hieronder staan de stakeholders die betrokken zijn bij dit project.

Nummer	Stakeholder
S1	Djimaro Talahatu(Afstudeerder)
S2	Eva de Schipper(Cito) (Opdrachtgever)
S4	Onderzoekers

Tabel 4.4: Stakeholders

Hieronder staat een lijst van concerns. Per stakeholder is er aangegeven welke zorgen de stakeholder heeft.

Nummer	Stakeholder	Concern
C1	S1	Hij wil het afstuderen halen.
C2	S4	De onderzoekers zijn gewend om in een bestand met kolommen te werken.
C3	S2 S3	De onderzoekers willen de kwaliteit van open vragen waarborgen.
C4	S4	De applicatie kan de eigenschappen van Nederlandse variabelen visualiseren.
C5	S4	De onderzoekers hebben een beperkte programmeerachtergrond. 10% kent Fortran, C++, C en Python. 75% kent R en SQL.
C6	S4	De onderzoekers willen gebruik kunnen maken van de beste Natural Language Processing oplossingen.
C7	S4	De applicatie moet geschreven zijn in een programmeertaal die 10% van de onderzoekers of meer kennen.
C8	S4	Er zijn 10 gebruikers tegelijk.
C9	S4	Er kunnen 15.000-300.000 variabelen worden ingeladen.
C10	S4	Er is maar één versie van de applicatie.
C12	S4	De onderzoekers willen de eigenschappen van de applicatie kunnen uitbreiden, aanpassen, toevoegen, verwijderen zonder daar een nieuwe programmeertaal voor te leren.
C13	S4	Als er tijdens het wachten iets fout gaat moet er een melding worden gegeven.
C14	S4	De applicatie mag niet te langzaam zijn.
C15	S4	De applicatie moet op verschillende locaties te gebruiken zijn.

Tabel 4.6: Concerns

5 Architecturally Significant Requirements

Requirement in software is een eis waar het systeem aan moet voldoen.

De architectuur van een systeem is een set van structuren die nodig zijn om te redeneren over het systeem. (Fairbanks, 2013)

Architectural significant requirements zijn requirements die invloed hebben op de architectuur. Deze requirements beïnvloeden de architectuur van de applicatie. De volgende requirements beïnvloeden de applicatie:

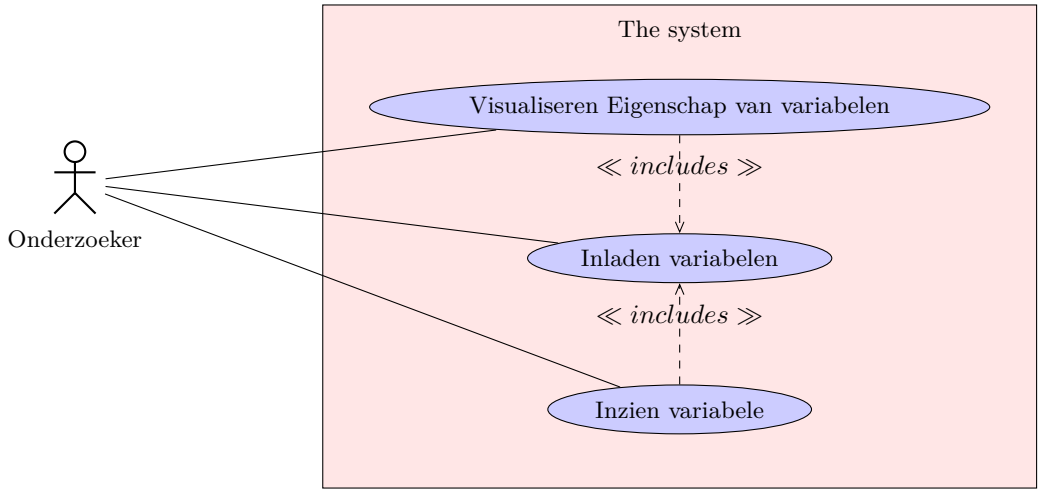
- High level use cases
- Quality attribute requirements
- External interfaces
- Technical constraints

Al deze requirements beïnvloeden de structuur of het gedrag van het systeem. In dit hoofdstuk worden deze requirements besproken.

5.1 High-level Use Cases

Het doel van een use case is een vaststelling van wat de gebruiker met de use case hoopt te bereiken. (Hoogendoorn, 2004, p. 70). Voor een gedetailleerde uitleg over de use cases (Zie Bijlage F) zie hoofdstuk use cases.

In dit gedeelte staat een overzicht van high level use cases. Deze use cases hebben het gedrag en structuur van de applicatie beïnvloed. Deze use cases zijn gevonden tijdens het ontwerpen van de applicatie.



Figuur 5.2: Overview

5.1.1 Use case 1: Inladen variabelen

De onderzoeker wil een bestand met variabelen inladen om deze later te analyseren of te vergelijken. Onderzoeker selecteert bestand. Het systeem laadt het bestand in. De gebruiker kan andere use cases uitvoeren.

5.1.2 Use case 2: Visualiseren Eigenschap van variabelen

De onderzoeker wil een eigenschap van variabelen visualiseren. Hij kiest de gewenste eigenschap en welke variabelen hij wil visualiseren. Het systeem geeft inzicht in de eigenschap van de gewenste variabelen.

5.1.3 Use case 3: Inzien variabele

De onderzoeker wil een variabele inzien. De onderzoeker selecteert de gewenste variabele. Het systeem toont de variabelen.

5.2 Quality Attribute Requirements

Een quality attribute is een meetbare of testbare eigenschap van het systeem dat aantoont hoe goed het systeem voldoet aan de eisen van de stakeholders.(Bass et al., z.d., p. 63).

Een quality attribute requirement is een eigenschap van een use case of van het hele systeem.(Bass et al., z.d., p. 64) Een eigenschap van een use case is bijvoorbeeld hoe snel iets wordt uitgevoerd. Een eigenschap van het hele systeem is bijvoorbeeld hoeveel resources het mag kosten.

Deze quality attribute requirements komen van de stakeholders. Deze requirements zijn achterhaald met behulp van use cases en schermontwerpen. Voor de usecases (Zie Bijlage F) hoofdstuk use cases. Voor de schermontwerpen (Zie Bijlage F) hoofdstuk User interface sketches.

Hier staat een overzicht van alle quality attributes requirements en de bijbehorende concerns.

Nr	Concern	Requirement	Quality attribute	Beschrijving
QAR1	C2	De applicatie laadt variabelen die in een bestand met kolommen en rijen staan in.	Functional completeness	
QAR2	C3C4	De applicatie moet eigenschappen van Nederlandse teksten kunnen visualiseren.	Functional completeness	Deze requirement is afhankelijk van de scripts die onderzoekers schrijven. Hierom wordt er geen rekening gehouden met deze requirement.
QAR3	C3C4	De applicatie kan eigenschappen van variabelen op niveau vergelijken.	Functional completeness	Er wordt geen rekening gehouden met deze requirement, omdat het tonen van eigenschappen via een figuur wordt gedaan. Het per niveau tonen is dus afhankelijk van het script die het figuur maakt.
QAR4	C3	Geeft de mogelijkheid om variabelen te inspecteren.	Functional completeness	
QAR5	C10 C12	De applicatie is uit te breiden door een script toe te voegen aan de repository.	Adaptability	
QAR6	C5 C10	Bij het verwijderen van één script wordt deze verwijderd uit de repository.	Adaptability	
QAR7	C5 C10 C12	Na het aanpassen, toevoegen of verwijderen van één of meerdere scripts, wordt deze geüpdatet voor alle gebruikers zodra de applicatie opnieuw wordt opgestart.	Modifiability	
QAR8	C13	Als er een fout optreedt tijdens het inladen, dan moet er een foutmelding worden weergegeven.	Fault tolerance	
QAR9	C13	Als er een fout optreed tijdens het achterhalen van eigenschappen dan moet er een foutmelding worden weergegeven.	Fault tolerance	

QAR10	C14	Het inladen van variabelen mag maximaal $O(n^2)$ *duren.*	Time behaviour	
QAR11	C14	Het filteren van variabelen mag maximaal $O(n)$ duren.	Time behaviour	
QAR12	C14	Het achterhalen van complexe eigenschappen van tekstuele variabelen mag voor een set van 15000 variabelen met 500 woorden maximaal 45 minuten duren.	Time behaviour	Wordt geen rekening mee gehouden, omdat dit afhangt van hoe de script is geschreven.
QAR13	C14	Het achterhalen van complexe eigenschappen van tekstuele variabelen mag voor een set van 15000 variabelen met 150 woorden maximaal 15 minuten duren.	Time behaviour	Wordt geen rekening mee gehouden, omdat dit afhangt van hoe de script is geschreven.
QAR14	C14	De applicatie moet normaal functioneren op een laptop met een 8'ste generatie quad core i5 met 8gb werkgeheugen.	Resource utilization	
QAR15	C8 C14	Als er minder dan 3000 rijen aan variabelen in het bestand staan dan wordt het achterhalen van de eigenschappen op de laptop uitgevoerd.	Capacity	
QAR16	C8 C14	Als er meer dan 3000 rijen aan variabelen in het bestand staan dan wordt het achterhalen van de eigenschappen op de server uitgevoerd.	Capacity	
QAR17	C9	De applicatie heeft de mogelijkheid om 15.000 - 300.000 variabelen in te laden.	Adaptability	
QAR18	C13	Als er een fout optreed tijdens het inladen of achterhalen van eigenschappen dan moet de gebruiker handmatig het proces opnieuw starten.	Operability	
QAR19	C15	De applicatie moet makkelijk te installeren zijn.	Installability	
QAR20	C15	De applicatie kan met internet onafhankelijk van locatie gebruikt worden.	Installability	
QAR21	C15	Achterhaald de eigenschappen van de variabele op basis van een R of Python script die een onderzoeker heeft geschreven.	Installability	

QAR22	C15	Bij het aanpassen van een script wordt deze aangepast in de repository.	Adaptability	Door een script te verwijderen en een script toe te voegen wordt dit al gerealiseerd.

Tabel 5.7: Quality attributes en concerns

5.2.1 Quality Attribute Scenario's

Om te controleren of een quality attribute requirement in het systeem zit, wordt dit getest of gemeten. Dit is nodig om te garanderen dat de applicatie werkt zoals hij hoort te werken.

Deze controle gebeurt door middel van quality attribute scenario's. Per scenario wordt er één quality attribute requirement getest. Deze scenario's worden dan gebruikt om aan te tonen dat architecturale requirements in de applicatie zitten.

Hieronder staat een uitleg over de quality attribute scenario's (Bass et al., z.d., p. 176). Per onderdeel wordt uitgelegd wat het is.

- Source: Een eindgebruiker.
- Stimulus: Een actie van een eindgebruiker die het systeem gebruikt.
- Artifact: Het systeem of een gedeelte van het systeem waarmee de gebruiker interactie heeft.
- Environment: Wanneer de actie wordt uitgevoerd. Dit kan tijdens het uitvoeren of het configureren van de applicatie.
- Response: Het systeem dat de gebruiker voorziet van functionaliteiten of wat de gebruiker nodig heeft.
- Response measure: De response die meetbaar is.

Hieronder staat een overzicht van de quality attribute scenario's. Deze zijn opgedeeld per use case en overig.

5.2.2 Inladen variabelen

Nr	QAR	Source	Stimulus	Artifact	Environment	Response	Response Measure
	QAR1 QAR10	Onderzoekers	De onderzoeker wil een bestand uploaden.	Het systeem	Normale operatie	Het systeem laadt het bestand in en geeft aan wanneer het inladen klaar is.	Het inladen duurt maximaal 2 minuten.
	QAR8	Onderzoekers	De onderzoeker wil een bestand met fouten erin uploaden.	Het systeem	Normale operatie	Het systeem probeert het bestand in te laten en gaat fout.	Het toont een foutmelding dat het inladen fout is gegaan.
	QAR11	Onderzoekers	Het systeem geeft aan de hand van filter criteria aan welke responses er gebruikt worden.	Het systeem	Normale operatie	Het systeem filtert de responses op basis van filter criteria.	Het systeem heeft criteria gefilterd.
	QAR18	Het systeem	Het systeem achterhaalt de eigenschappen van de responses	Het systeem	Normale operatie	Er gaat iets fout in de script waarmee de eigenschappen worden achterhaald.	Het toont een foutmelding dat zegt dat het achterhalen van eigenschappen fout is gegaan.

Tabel 5.8: Quality Attribute Scenario's

5.2.3 Visualiseren Eigenschap van variabelen

QAR	Source	Stimulus	Artifact	Environment	Response	Response Measure
QAR2	Onderzoekers	De onderzoeker wil teksteigenschappen van responses visualiseren.	Het systeem	Normale operatie	Het systeem visualiseert de teksteigenschappen van de responses.	Het systeem toont de eigenschappen door een afbeelding te tonen.

QAR9	Onderzoekers	De onderzoeker wil teksteigenschappen van responses visualiseren.	Het systeem	Normale operatie	Er zit een fout in het script.	Het systeem geeft aan dat het achterhalen fout is gegaan en bij welke eigenschap het fout is gegaan.
QAR15	Onderzoekers	De onderzoeker wil de eigenschappen van responses achterhalen. De applicatie en responses kosten meer geheugen dan de laptop aankan.	Het systeem	Normale operatie	De applicatie achterhaalt de eigenschappen op de server.	De server wordt gebruikt.
QAR16	Onderzoekers	De onderzoeker wil eigenschappen van responses achterhalen. De applicatie en responses kosten minder geheugen dan de laptop aankan.	Het systeem	Normale operatie	Applicatie achterhaald de eigenschappen op de laptop.	De laptop wordt gebruikt.
QAR5	Onderzoekers	De onderzoeker wil een nieuw script toevoegen.	Het systeem	Normale operatie	Het script is toegevoegd aan de repository.	
QAR6 QAR7	Onderzoekers	De onderzoeker wil een script in de repository verwijderen.	Het systeem	Normale operatie	Het script is verwijderd uit de repository.	
QAR22	Onderzoekers	De onderzoeker wil een script in de repository aanpassen.	Het systeem	Normale operatie	Het script is aangepast.	
QAR7	Onderzoekers	De onderzoeker start de applicatie op.	Het systeem	Normale operatie	De applicatie haalt scripts op uit de repository.	De scripts zijn opgehaald uit de repository.

Tabel 5.9: Quality Attribute Scenario's

5.2.4 Inzien variabele

QAR	Source	Stimulus	Artifact	Environment	Response	Response Measure
QAR1	Onderzoekers	De onderzoeker wil een bestand uploaden.	Het systeem	Normale operatie	Het systeem laadt het bestand in en geeft aan wanneer het inladen klaar is.	Het inladen duurt maximaal 2 minuten

Tabel 5.10: Quality Attribute Scenario's

5.3 External Interfaces

Een external interface maakt het mogelijk dat twee losstaande applicaties met elkaar kunnen samenwerken. Hier worden de applicaties besproken die nodig zijn om deze applicatie te laten functioneren.

Deze applicaties zijn achterhaald door de requirements te analyseren en te redeneren over het systeem. Voor de requirements Zie hoofdstuk 5.2 op pagina 9. Voor het redeneren is er gebruik gemaakt van diagrammen. Voor de gebruikte diagrammen zie hoofdstuk 6 op pagina 16.

Deze applicatie maakt gebruik van github (Github, z.d.). Hiervoor is er een beslissing gemaakt. Deze beslissing is te vinden op pagina 30.

5.4 Technical constraints

Technical constraints zijn architecturale beslissingen die van te voren gemaakt zijn en invloed hebben op de structuur van de applicatie.

Deze constraints zijn achterhaald door de requirements te analyseren. Voor de requirements zie hoofdstuk 5.2 op pagina 9. Daarnaast is er ook geredeneerd over het systeem. Hierbij is er gebruik gemaakt van diagrammen. Zie hoofdstuk 6 op pagina 16 voor de gebruikte diagrammen.

Bij het ontwikkelen van de applicatie moet er gebruik worden gemaakt van een taal waarmee de onderzoekers bekend zijn.

6 Architectural Views

De onderzoekers van Cito zijn stakeholders met verschillende achtergronden. Vanwege de verschillende achtergronden heeft elke onderzoeker andere zorgen bij deze applicatie.

De afstudeerder is een stakeholder met het profiel software development. De zorgen van de afstudeerder zijn daarom gerelateerd aan software development.

Sommige architecturen worden weergegeven in één diagram. Een architectuur diagram beschrijft hoe de applicatie is opgedeeld, hoe het omgaat met fouten, het gedrag van de applicatie en nog meer.

Als er maar één architectuur diagram is, moeten de zorgen van de stakeholders erin staan die te maken hebben met de architectuur. De stakeholders hebben namelijk zorgen bij de applicatie en hebben duidelijkheid nodig. Als er verschillende stakeholders zijn bevat de diagram meer details. Bij dit project zijn er verschillende stakeholders met verschillende achtergronden. Elke stakeholder heeft vanwege zijn achtergrond andere zorgen. Hierdoor is er bij het gebruik van één diagram meer details nodig. Er zijn details nodig om duidelijk te maken hoe de zorgen zijn aangepakt. Als er veel verschillende stakeholders zijn, bevat het diagram teveel details en wordt het diagram onduidelijk. Een stakeholder ziet een diagram en de details. Bij de details raakt hij afgeleid, omdat er details tussen zitten die niet belangrijk zijn voor hem. Omdat er stakeholders zijn met verschillende achtergronden is er meer dan één diagram en verschillende diagrammen nodig om de architectuur duidelijk te maken. Doordat er verschillende soorten diagrammen zijn wordt er verwacht dat elke stakeholder een duidelijk overzicht krijgt. Hierdoor wordt het voor de stakeholders gemakkelijker om de applicatie te onderhouden.

4+1 is een model dat bestaat uit vier verschillende diagrammen voor verschillende stakeholders die gebaseerd zijn op use cases (Zie Bijlage G) An Architectural Model. Deze diagrammen worden verder in dit hoofdstuk behandeld. 4+1 bestaat uit de logical view, development view, proces view en deployment view An Architectural Model (Zie Bijlage G) An Architectural Model.

4+1 zorgt ervoor dat de architectuur duidelijk wordt weergegeven door vier verschillende diagrammen. Voor de use cases die te maken hebben met de architectuur zie hoofdstuk 5.1 op pagina 8. Voor alle use cases (Zie Bijlage F) hoofdstuk use case descriptions.

Door 4+1 hebben de stakeholders meer inzicht over wat de applicatie kan. Hierom is gekozen voor 4+1. Door dit model krijgen de stakeholders meer duidelijkheid en dus meer inzicht. Hierdoor kunnen de ontwikkelaars en stakeholders beter samenwerken.

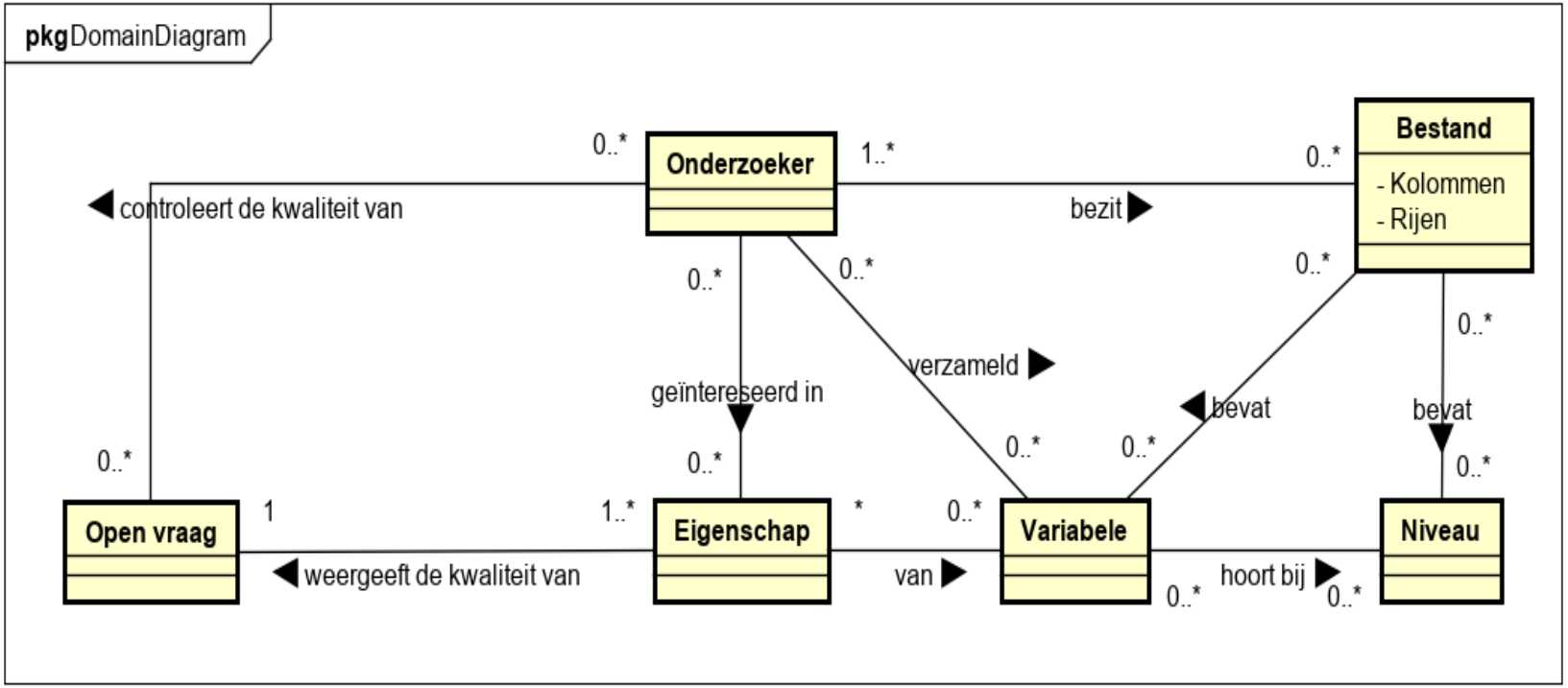
De diagrammen van 4+1 zijn gemaakt op basis van de requirements en use cases. Zie hoofdstuk 5.2 op pagina 9 voor de requirements. (Zie Bijlage F) hoofdstuk use case descriptions voor de use cases.

De beslissingen zijn verder in dit hoofdstuk terug te vinden. Daar worden de beslissingen die horen bij één van de diagrammen individueel besproken, in een overzicht getoond, de relaties tussen beslissingen worden getoond en wordt besproken op welke volgorde de beslissingen zijn gemaakt.

6.1 Logical View

De logical view beschrijft het systeem in de vorm van objecten.(Zie Bijlage G) An Architectural Model.

Op basis van het proces om de kwaliteit van een open vraag te bepalen en hun werkwijze en terminologie dat de opdrachtgever heeft beschreven is de logical view gemaakt. Zie onderstaand figuur voor de logical view.

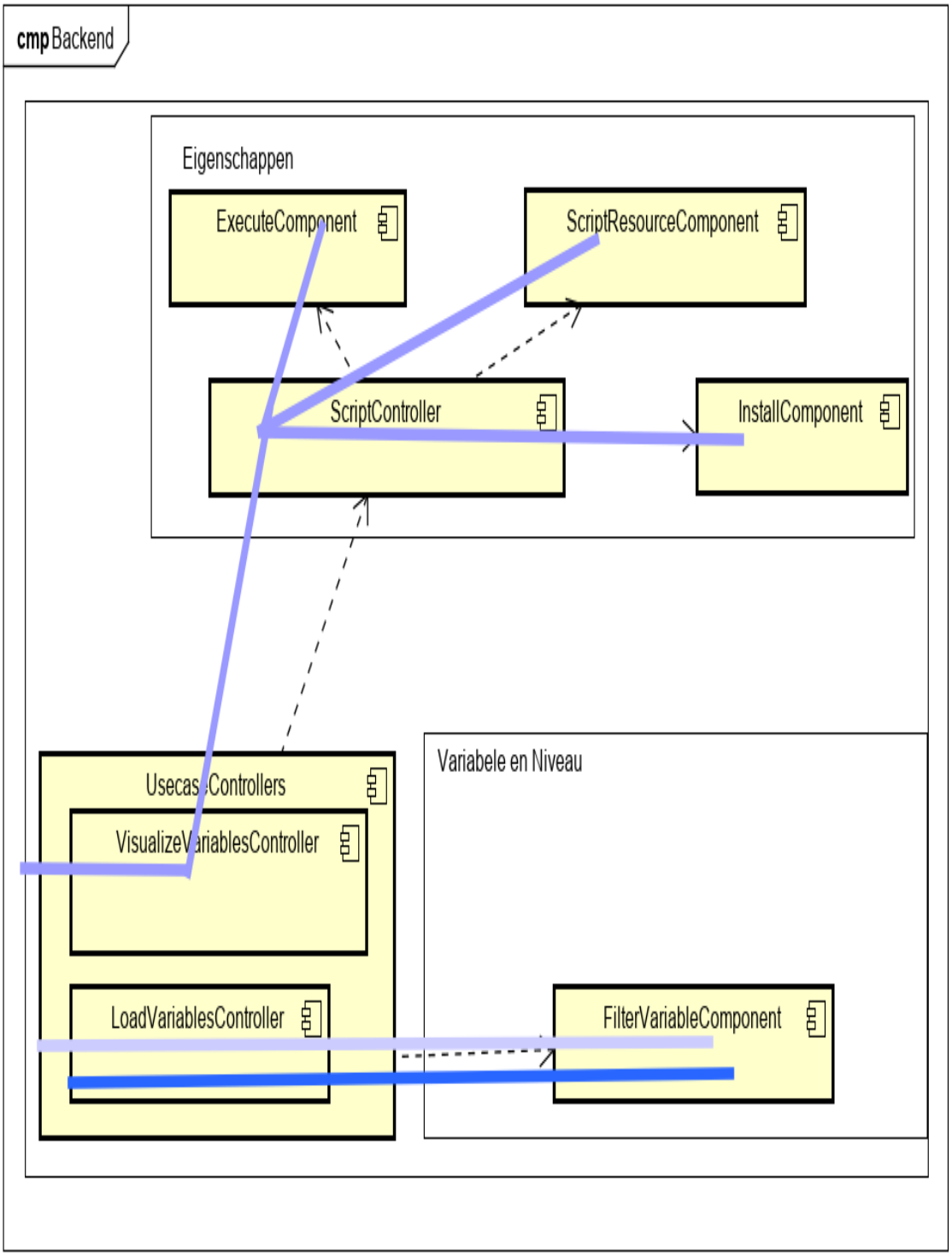


Figuur 6.3: Domain model

6.2 Development view

De development view beschrijft hoe de software is georganiseerd.(Zie Bijlage G) An Architectural Model.

Deze software is gegroepeerd op basis van onderdelen die waarschijnlijk tegelijkertijd veranderen. Deze onderdelen komen uit de use cases (Zie Bijlage F) hoofdstuk use case descriptions, terminologie (Zie hoofdstuk 1) en het domain model (Zie Bijlage F) hoofdstuk domain model.



Figuur 6.4: Development view

Kleur	Usecase
	Inladen variabelen
	Visualiseren Eigenschap van variabelen
	Inzien variabele

UsecaseControllers:
UsecaseController is het begin van de backend van het programma. Hiermee kunnen de volgende usecases worden uitgevoerd:

- Inladen variabelen
- Visualiseren Eigenschap van variabelen

FilterVariableComponent:
Hier wordt er op basis van de niveaus de responses gefilterd.

ScriptController:
Hier kan alles gedaan worden wat met scripts te maken heeft.

ExecuteComponent:
Hier worden de scripts uitgevoerd om de eigenschappen van variabelen te achterhalen.

ScriptResourceComponent:
Hier worden de scripts geupdate en gedownload uit de repository.

InstallComponent:
Voordat een script wordt uitgevoerd, moeten de juiste libraries worden geinstalleerd. Dat gebeurt hier.

6.3 Process View

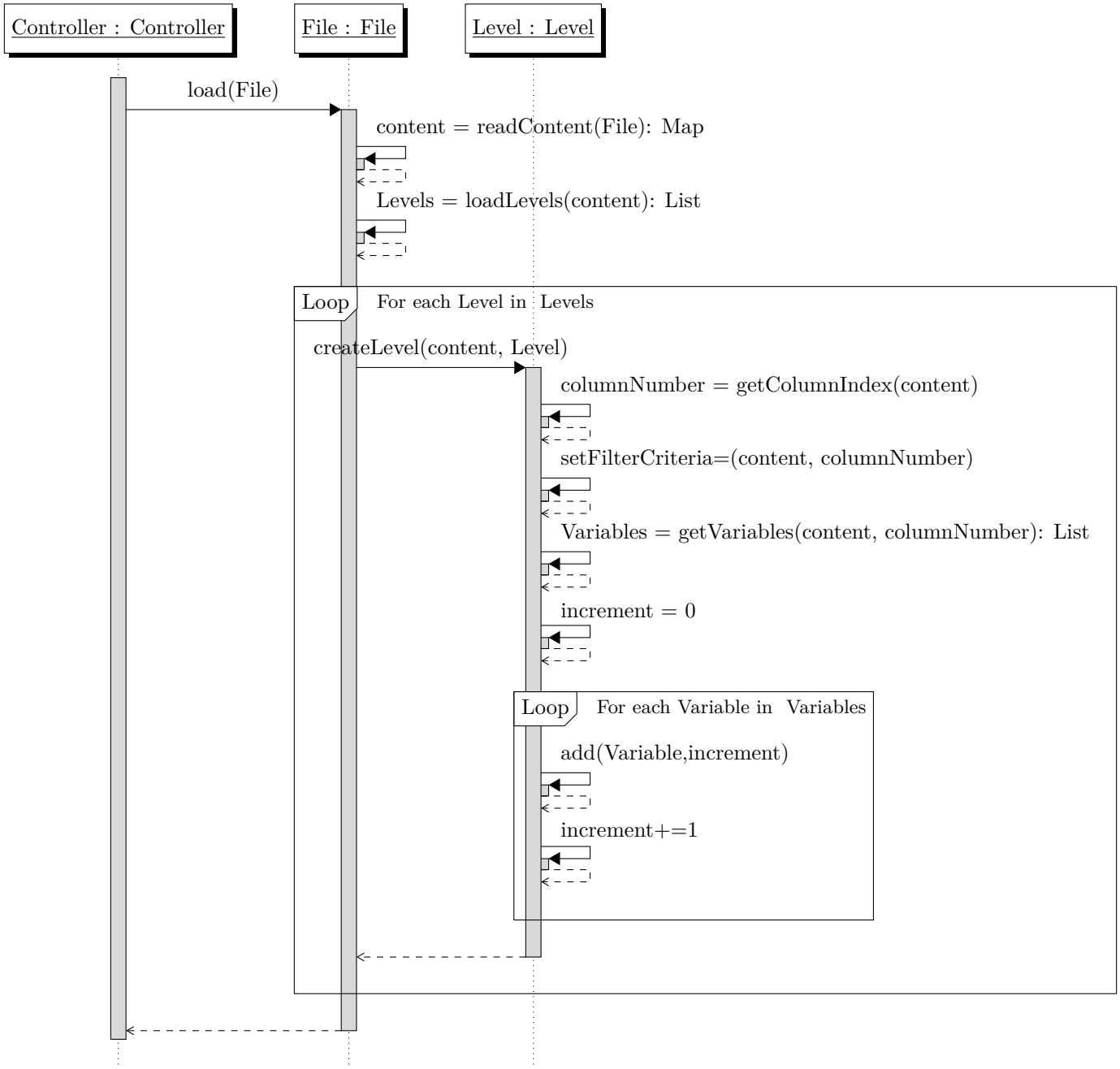
De process view geeft weer hoe het systeem werkt. (Zie Bijlage G) An Architectural Model.

Hier wordt er per use case aangegeven hoe het systeem werkt. Per diagram is er aangegeven met welke requirements er rekening gehouden zijn. Op basis van de requirements zijn deze process views dan ook gemaakt.

Het filteren wordt gebruikt bij de volgende use cases:

- Visualiseren Eigenschap van variabelen
- Inzien variabele

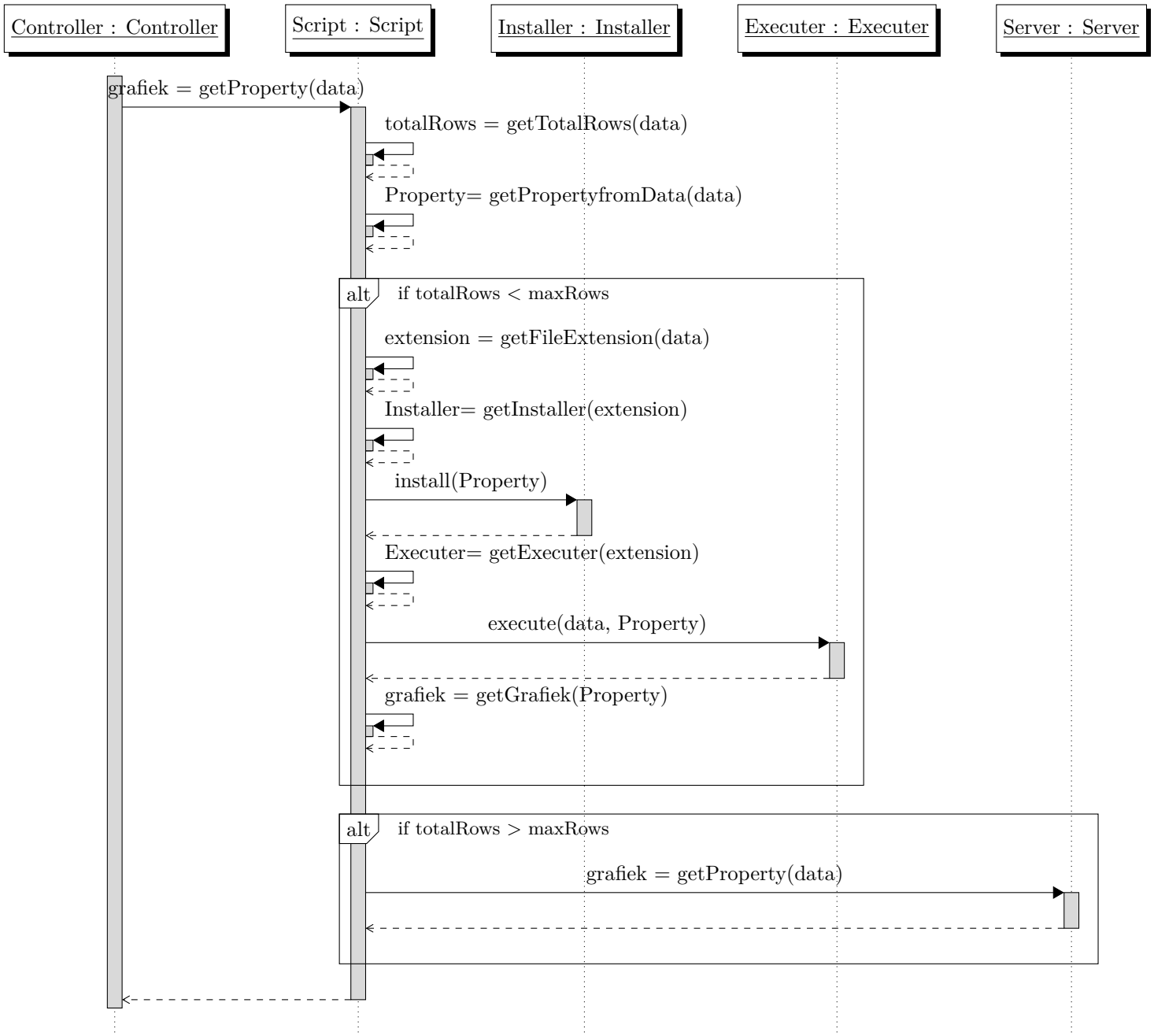
6.3.1 Inladen variabelen



Hieronder staat een overzicht van de requirements waarmee rekening is gehouden.

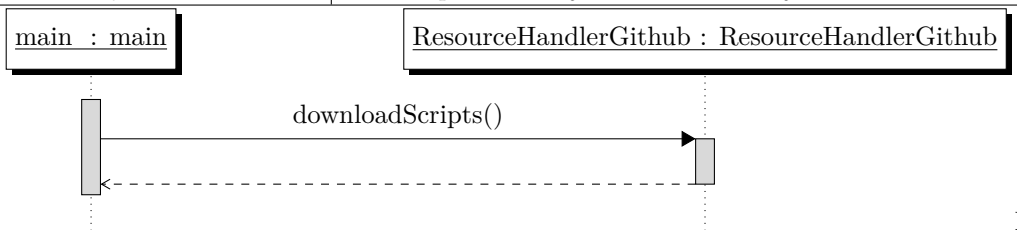
Quality attribute requirement	Beschrijving
QAR1	readContent() en loadLevels() zorgen ervoor dat de applicatie gemakkelijk uit te breide
QAR8	Als er iets fout gaat tijdens het inladen, wordt er een
QAR10	Het duurt iets langer dan $n^2, omdatdevariabelenwordeninge$
QAR11	Bij het inladen worden de variabelen ingeladen in een AVLTree. De AVLTree zorgt ervoor en gara

6.3.2 Visualiseren Eigenschap van variabelen



Hieronder staat een overzicht van de requirements waarmee rekening gehouden is.

Quality attribute requirement	Beschrijving
QAR9	Er wordt een foutmelding getoond doorgegeven als er een fout optreedt.
QAR15	Het script wordt bij 3000 rijen aan variabelen op de laptop uitgevoerd.
QAR16	Het script wordt bij meer dan 3000 rijen aan variabelen op de server uitgevoerd.



Hieronder staat een overzicht van de requirements waarmee rekening is gehouden.

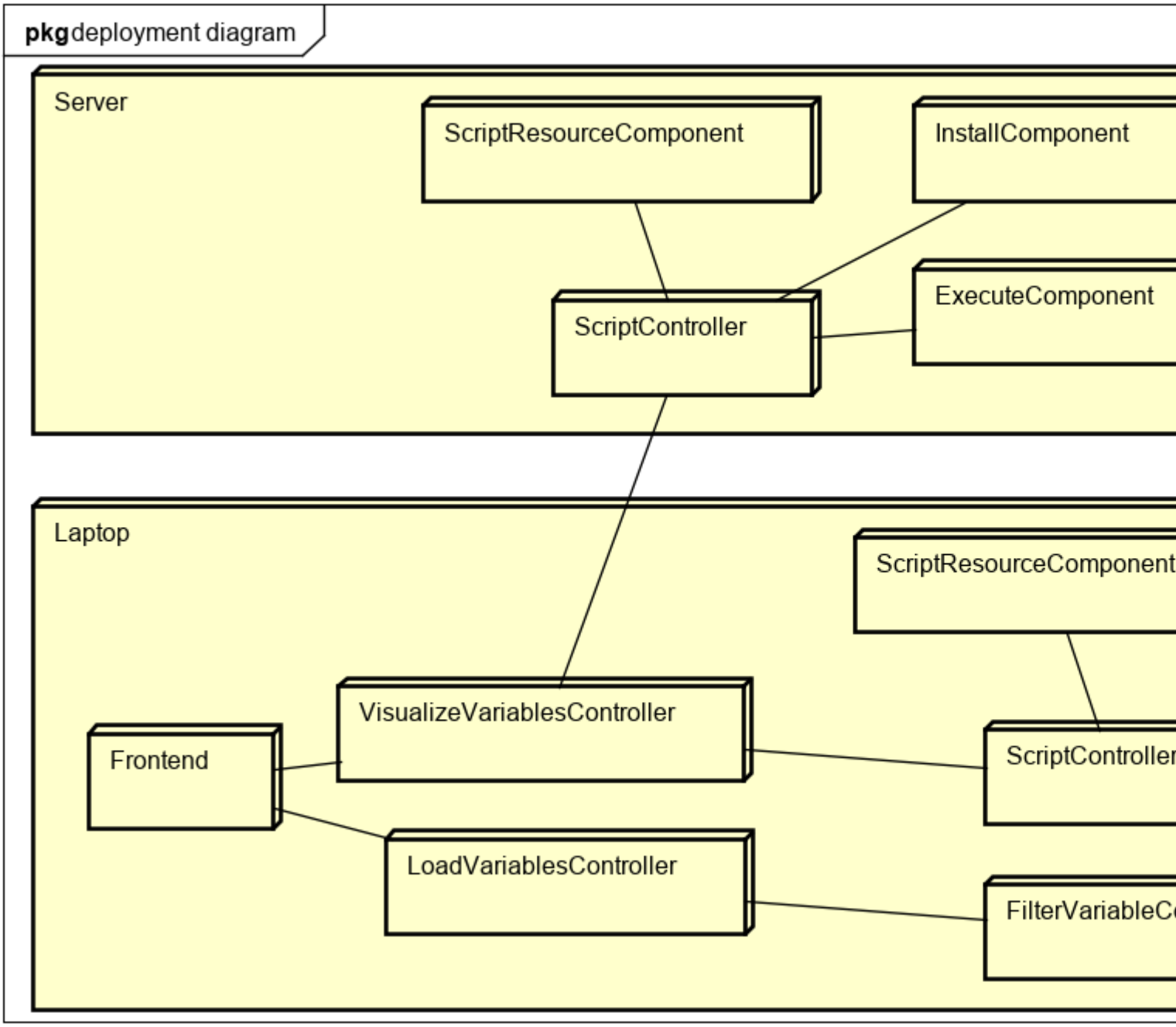
Quality attribute requirement	Beschrijving
QAR7	De scripts worden gedownload nadat de applicatie is opgestart.

6.4 Deployment view

De deployment view geeft weer waar de software wordt uitgevoerd. (Zie Bijlage G) An Architectural Model.

Deze view is gemaakt op basis van requirements Zie hoofdstuk 5.2 op pagina 9, concerns, inzichten van de afstudeerder, de development view en process views.

architecture document/Figuren/Deployment Diagram0.png architecture document/Figuren/Deployment Diagram0.png



Figuur 6.5: Deployment view

7 Decisions

Bij het maken van een architectuur worden er beslissingen genomen om de problemen op te lossen. Hierbij worden de meest waarschijnlijke problemen als eerst opgelost, omdat er gebruik wordt gemaakt van RUP. (Rupopmaat, z.d.) Vanuit HAN is er een template beschikbaar gesteld voor het documenteren van beslissingen. Deze zijn terug te vinden in het subhoofdstuk Decision detail view. Zie 7.1 op pagina 26.

Om een probleem op te lossen moeten de beslissingen ervan onderbouwd worden. De onderbouwing hangt af van verschillende factoren. Om deze factoren inzichtelijk te maken is er vanuit HAN een template beschikbaar gesteld. Dit overzicht is terug te vinden in Decision forces view (Zie Bijlage E).

Tijdens het onderbouwen van een beslissing kunnen oude beslissingen de uitkomst ervan beïnvloeden. Hiervoor heeft de HAN een template beschikbaar gesteld. Omdat dit document is geschreven in Latex is er een afgeleide versie van gemaakt. De geschiedenis van alle beslissingen is terug te vinden in subhoofdstuk Chronological view. Zie hoofdstuk 7.4 op pagina 30.

Naast het onderbouwen moet er mogelijk rekening gehouden worden met andere beslissingen, omdat een beslissing impact kan hebben op een andere beslissing of andere beslissingen kan veroorzaken. Ook hier heeft HAN een template beschikbaar gesteld. Omdat dit document is geschreven in Latex is er een afgeleide versie van gemaakt. De decision relationship view is een overzicht van de relaties tussen elke beslissing. Zie hoofdstuk 7.3 op pagina 30.

Om te garanderen dat de beslissingen passend zijn bij de applicatie, moeten de beslissingen goedgekeurd worden door de stakeholder(s). Bij het definitief maken van een beslissing worden de relevante stakeholders betrokken om een besluit te nemen. De status van een beslissing zijn terug te vinden in de Decision detail zie hoofdstuk 7.1 op pagina 26 en Zie hoofdstuk 7.3 op pagina 30.

7.1 Decision Detail View

Een decision detail view geeft de beslissing weer in detail. Hier staan alle individuele beslissingen die te maken hebben met de architectuur. Deze beslissingen staan niet in een specifieke volgorde.

Name	Welke programmeertalen worden er voor Natural Language Processing gebruikt?
State	Accepted
Decision group	
Problem	Welke programmeertaal/talen wordt/worden er gebruikt voor het achterhalen van de eigenschappen van tekstuele responses?
Decision	Python en R
Alternatives	Python of R

Arguments	Dit is de verdeling van programmeertalen die de onderzoekers van Cito kennen: • Fortran 10% • C++ 10% • C 10% • R 75% • Python 10% • SQL 75% Hierbij is het belangrijk dat de applicatie modulair wordt opgesteld. Hieraan voldoen de volgende programmeertalen: Fortran, C, C++, Python en R. Uit (Magnimindacademy, 2018-a) bleek dat Python mogelijkheden bevat voor Natural Language Processing en R wordt gebruikt om grote hoeveelheden data te analyseren. De overige talen worden niet genoemd in de bovenstaande links. Hierom worden deze ook niet overwogen. Er wordt verwacht dat Python veel ondersteuning heeft, omdat deze op nummer één staat. Python heeft de voorkeur voor de afstudeerder, omdat hij hiermee ervaring heeft. De onderzoekers moeten grote hoeveelheden data analyseren. Hier kan R voor worden gebruikt. Enkel en alleen Python geeft beperkingen voor de onderzoekers voor het analyseren van grote hoeveelheden data. Hierom zijn beide programmeertalen gekozen.
Related decisions	
Related requirements	Concerns: C1, C5, C6, C7en C12. Requirements: QAR5, QAR6en QAR7

Tabel 7.11: Definities

Name	Inladen bestand met variabelen in memory, op de hardeschijf of in een database?
State	Accepted
Decision group	
Problem	Vanaf waar worden de responses gefilterd?
Decision	In memory
Alternatives	In memory, Harde schijf en of Database

Arguments	In memory: • +: De variabelen zitten in het geheugen en zijn dus snel om te filteren. • -: Het is minder schaalbaar(Bij het inladen en filteren van video's gaat het waarschijnlijk fout, omdat deze te groot kunnen zijn om in het geheugen worden gehouden). • +: Het is simpeler om de applicatie te installeren. Harde schijf: • -: Op de harde schijf is filteren langzamer, omdat de schijf langzamer is dan het geheugen. • +: Het is schaalbaar: bij het inladen video's gaat het goed, omdat deze op een plek worden ingeladen waar genoeg ruimte is. Database: • -: In de database filteren is langzamer, omdat de data eigenlijk vanaf de harde schijf wordt gelezen. • +: Het is schaalbaar: bij het inladen video's gaat het goed, omdat deze op een plek worden ingeladen waar genoeg ruimte is. Op basis van de memory test <bijlage> is gebleken dat er meer dan 1 miljard woorden in het geheugen kunnen worden opgeslagen. Het kost de applicatie 7 gb aan geheugen. Dit is mogelijk doordat er gebruik is gemaakt van python 3 64 bit. Tekstuele responses kunnen in het geheugen worden ingeladen, omdat het geheugen sneller is dan de database of het uitlezen vanaf de harde schijf.
Related decisions	variabelen in bestand of buiten bestand
Related requirements	Concerns: C9en C14Requirements: QAR10, QAR11en QAR17

Tabel 7.12: Definities

Name	variabelen in bestand of buiten bestand
State	Accepted
Decision group	
Problem	Waar voeren de onderzoekers de responses op een schaalbare manier in?
Decision	variabelen in het bestand.
Alternatives	Responses via het bestand inladen.
Arguments	De onderzoekers zijn gewend om in een bestand met kolommen en rijen te werken. Hierom worden de variabelen ingeladen vanuit het bestand.
Related decisions	Inladen bestand met variabelen in memory, op de hardeschijf of in een database?
Related requirements	Concerns: C2en C9Requirements: QAR1en QAR17

Tabel 7.13: Definities

Name	In welke programmeertaal wordt de backend geschreven
State	Accepted
Decision group	
Problem	Welke programmeertaal wordt er gebruikt voor de backend?
Decision	Python
Alternatives	Fortran 10%, C++ 10% , C 10% , R 75%, Python 10% of SQL 75%
Arguments	Dit is de verdeling van programmeertalen die de onderzoekers van Cito kennen: • Fortran 10% • C++ 10% • C 10% • R 75% • Python 10% • SQL 75% Er moet gebruik worden gemaakt van een taal waarmee de onderzoekers bekend zijn.Er moet gebruik worden gemaakt van een taal die dynamisch is zodat de scripts kunnen worden geüpdatet terwijl de applicatie wordt uitgevoerd. Python en R worden al gebruikt in de applicatie voor het achterhalen van de response eigenschappen. De afstudeerder heeft op school al enige ervaring opgedaan met Python. De onderzoekers hebben niet zoveel ervaring met Python. Er zijn onderzoekers die de eigenschappen van responses achterhalen met behulp van Python. Er zijn onderzoekers die de eigenschappen van responses achterhalen met behulp van R. De afstudeerder heeft geen ervaring met R. Omdat Python voor de afstudeerder een lagere leercurve heeft, Python al gebruikt wordt bij scripts en Python dynamisch is, is er gekozen voor Python.
Related decisions	Welke programmeertalen worden er voor Natural Language Processing gebruikt?
Related requirements	Concerns:C1, C5en C7Requirements: QAR6en QAR7

Tabel 7.14: Definities

Name	Welke programmeertaal wordt er voor frontend gebruikt?
State	Accepted
Decision group	
Problem	Welk framework wordt er voor frontend gebruikt?
Decision	Angular
Alternatives	React of Vue

Arguments	De onderzoekers hebben geen ervaring met frontend. Er is een frontend nodig voor de onderzoekers die gebruik willen maken van de applicatie. De afstudeerder heeft enige ervaring met Angular. Vanuit Luminis zijn er veel mensen die ervaring hebben met Angular. Vanwege de ondersteuning die Luminis kan bieden en de ervaring, is er gekozen voor Angular.
Related decisions	Welke programmeertaal wordt er voor frontend gebruikt?
Related requirements	Concerns: C1, C5

Tabel 7.15: Definities

Name	Waar worden de scripts opgeslagen?
State	Accepted
Decision group	
Problem	Waar worden de scripts opgeslagen?
Decision	Github
Alternatives	Dropbox en Gitlab.
Arguments	Gitlab en GitHub zijn praktisch hetzelfde. De opdrachtgever en de afstudeerder hebben meer ervaring met GitHub. Dropbox is niet goed in file versioning als GitHub, omdat je maar één file per keer kan herstellen. Daarnaast is het lastiger om alreeds verwijderde files terug te halen. (Dropbox, 2019)
Related decisions	
Related requirements	Requirement: QAR14, QAR15, QAR16, QAR17

Tabel 7.16: Definities

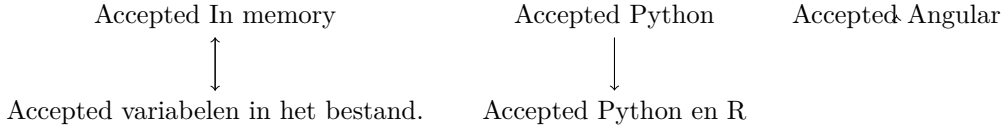
7.2 Decision-forces view

De decision forces view is een overzicht van factoren. In dit overzicht is aangegeven welke factoren bijdragen aan het maken van een beslissing. Voor een overzicht van de decision forces view (Zie Bijlage A).

7.3 Decision relationship View

De decision relationship view is een overzicht van de relaties tussen beslissingen. Deze view is gemaakt op basis van de beslissingen uit het vorige hoofdstuk. Voor alle beslissingen zie hoofdstuk 7.1 op pagina 26.

In het overzicht is er door middel van pijlen aangegeven welke beslissingen invloed op elkaar hebben. Hierbij staat de status als eerste aangegeven en daarna het resultaat van een beslissing.



7.4 Chronological View

Hieronder staat de geschiedenis van alle beslissingen. Deze beslissingen staan op volgorde. In sprint één is het projectplan gemaakt. In sprint twee is het onderzoek uitgevoerd. Dat is de reden waarom de view begint bij sprint drie.

Python en R

↓

variabelen in het bestand.

↓

Python

↓

In memory

↓

Angular

↓

Einde sprint 3

↓

Github

↓

Einde sprint 4

8 References

9 Bronnenlijst

Bass, L., Clements, P., & Kazman, R. (z.d.). Software Architecture in Practice (3e druk). Cambridge: Pearson.

Dropbox. (2019, 1 oktober). How to restore a previous file version | Dropbox Tutorials | Dropbox. Geraadpleegd op 6 april 2020, van https://www.youtube.com/watch?v=UPBBira72_Q

Fairbanks. (2013, 3 januari). What is Software Architecture? Geraadpleegd op 22 april 2020, van https://www.youtube.com/watch?v=Rn1g6V-vlHw

Github. (z.d.). Github. Geraadpleegd op 22 april 2020, van https://github.com/

Rupopmaat. (z.d.). RUP op Maat <naslagsite>. Geraadpleegd op 16 december 2019, van http://www.rupopmaat.nl/naslagsite2011/index.html

Magnimindacademy. (2019, 26 oktober). What programming languages are suitable for Natural Language Processing? . Geraadpleegd op 6 april 2020, van https://magnimindacademy.com/what-programming-languages-are-suitable-for-natural-language-processing/

Hoogendoorn, S. (2004). USE CASES opstellen en testen. Cambridge: Pearson.

10 Bijlagen

Bijlage A : Schrijftaak
Bijlage B : AntwoordenLeerlingen
Bijlage C : Onderzoek
Bijlage D : Interview
Bijlage E : DecisionForcesView
Bijlage F : Functioneel ontwerp
Bijlage G : Kruchten$_4$1$ViewModel$