HO
GENT

Faculteit Bedrijf en Organisatie

Good-practice geluidsgeneratie en -verwerking

Victor Van Weyenberg

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Stefaan Samyn
Co-promotor:
Thomas Desmedt

Instelling: —

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Good-practice geluidsgeneratie en -verwerking

Victor Van Weyenberg

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Stefaan Samyn
Co-promotor:
Thomas Desmedt

Instelling: —

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

1 Inleiding 15

1.1 Probleemstelling 15

1.2 Afbakening 15

1.2.1 In-scope 16

1.2.2 Out-of-scope 16

1.3 Onderzoeksvragen 16

1.4 Onderzoeksdoelstelling 18

1.5 Opzet van deze bachelorproef 18

2 Stand van zaken 19

2.1 Inhoud 19

2.2 Interviews 19

2.2.1 Correspondentie 19

2.2.2 Bart Vincent (Producer) 20
2.2.3 Peter Boone (Producer en Muzikaal Artiest) 21
2.2.4 Thomas Houthave (Sound Designer) 22
2.2.5 Vagabundos (Band) 23
2.3 Methodes voor Geluidsgeneratie en -verwerking 24
2.3.1 Subtractive Synthesis 24
2.3.2 Additive Synthesis 25
2.3.3 Granular Synthesis 25
2.3.4 Wavelet en Corpus-based Synthesis 26
2.4 Libraries 26
2.4.1 JASS 26
2.4.2 Beads 27
2.4.3 JSyn 28
3 Methodologie 31
3.1 Inhoud 31
3.2 Empirische tests 32
3.2.1 De Testmachine 32
3.2.2 De Code 32
3.2.3 Verwerking van de Testresultaten 36
3.3 Emotionele tests 37
4 Verloop van het Onderzoek 39
4.1 Is het mogelijk om digitaal te gaan? 39
4.1.1 Geluidsfilters 39

4.1.2 Harmonisch rijke golven 40

4.2 In het nuttig om digitaal te gaan? 41

4.2.1 Is het mogelijk om digitaal te gaan? 41

4.2.2 Wie heeft baat bij een digitalisering? 41

4.2.3 Zou een digitalisering verwelkomd worden? 41

4.3 Is het realistisch om digitaal te gaan? 41

5 Conclusie 43

5.1 Toekomstig Onderzoek 43

A Onderzoeksvoorstel 45

B Correspondentie 49

C Transcripties van de interviews 53

C.1 Bart Vincent (Producer) 53

C.2 Peter Boone (Producer en Muzikaal Artiest) 56

C.3 Thomas Houthave (Sound Designer) 59

C.4 Vagabundos (Band) 63

D Code van de Empirische Tests 71

E Testresultaten 83

E.1 Testresultaten van Beads 83

E.2 Testresultaten van JASS 90

E.3 Testresultaten van JSyn 97

F Smoothing van de Testresultaten 105

F.1 Smoothing van CPU testresultaten voor Beads 105

F.2 Smoothing van Mem testresultaten voor Beads 105

F.3 Smoothing van CPU testresultaten voor JASS 105

F.4 Smoothing van Mem testresultaten voor JASS 105

F.5 Smoothing van CPU testresultaten voor JSyn 105

F.6 Smoothing van Mem testresultaten voor JSyn 105

Lijst van figuren

3.1 Klasse diagram van de Template klasse 34
3.2 Linux commando om performantie van een proces te meten. . . . 35
3.3 *Moving median* formule 36
3.4 Hanning formule 36

4.1 Java blokgolf generatie functie 40
4.2 Smoothing van CPU testresultaten van testcase 4 voor JSyn 42

F.1 Smoothing van CPU testresultaten van testcase 1voor Beads 106
F.2 Smoothing van CPU testresultaten van testcase 2voor Beads 106
F.3 Smoothing van CPU testresultaten van testcase 3voor Beads 107
F.4 Smoothing van CPU testresultaten van testcase 4voor Beads 107
F.5 Smoothing van CPU testresultaten van testcase 5voor Beads 108
F.6 Smoothing van Mem testresultaten van testcase 1voor Beads . . . 108
F.7 Smoothing van Mem testresultaten van testcase 2voor Beads . . . 109
F.8 Smoothing van Mem testresultaten van testcase 3voor Beads . . . 109
F.9 Smoothing van Mem testresultaten van testcase 4voor Beads . . . 110
F.10 Smoothing van Mem testresultaten van testcase 5voor Beads . . 110
F.11 Smoothing van CPU testresultaten van testcase 1voor JASS 111

F.12 Smoothing van CPU testresultaten van testcase 2voor JASS 111
F.13 Smoothing van CPU testresultaten van testcase 3voor JASS 112
F.14 Smoothing van CPU testresultaten van testcase 4voor JASS 112
F.15 Smoothing van CPU testresultaten van testcase 5voor JASS 113
F.16 Smoothing van Mem testresultaten van testcase 1voor JASS 113
F.17 Smoothing van Mem testresultaten van testcase 2voor JASS 114
F.18 Smoothing van Mem testresultaten van testcase 3voor JASS 114
F.19 Smoothing van Mem testresultaten van testcase 4voor JASS 115
F.20 Smoothing van Mem testresultaten van testcase 5voor JASS 115
F.21 Smoothing van CPU testresultaten van testcase 1voor JSyn 116
F.22 Smoothing van CPU testresultaten van testcase 2voor JSyn 116
F.23 Smoothing van CPU testresultaten van testcase 3voor JSyn 117
F.24 Smoothing van CPU testresultaten van testcase 4voor JSyn 117
F.25 Smoothing van CPU testresultaten van testcase 5voor JSyn 118
F.26 Smoothing van Mem testresultaten van testcase 1voor JSyn 118
F.27 Smoothing van Mem testresultaten van testcase 2voor JSyn 119
F.28 Smoothing van Mem testresultaten van testcase 3voor JSyn 119
F.29 Smoothing van Mem testresultaten van testcase 4voor JSyn 120
F.30 Smoothing van Mem testresultaten van testcase 5voor JSyn 120

Lijst van tabellen

2.1 Vergelijking van de drie testlibraries. 29

3.1 Specificaties van de Testmachine 32
3.2 Testcase index per profiel 33
3.3 Testcase parameters per index 33

4.1 Medianen van de verwerkte testresultaten per library per testcase. 42

B.1 Lijst van correspondenten en hun functies. 50
B.2 Lijst van correspondentie met datum, medium, beschrijving en resultaat van het bericht. 52

C.1 Deelnemers van het interview en vernoemde namen. 63

E.1 Beads testresultaten van case 1. 83
E.2 Beads testresultaten van case 2. 84
E.3 Beads testresultaten van case 3. 85
E.4 Beads testresultaten van case 4. 87
E.5 Beads testresultaten van case 5. 88
E.6 Beads testresultaten van case 6. 89

E.7 JASS testresultaten van case 1. 90
E.8 JASS testresultaten van case 2. 91
E.9 JASS testresultaten van case 3. 92
E.10 JASS testresultaten van case 4. 94
E.11 JASS testresultaten van case 5. 95
E.12 JASS testresultaten van case 6. 96
E.13 JSyn testresultaten van case 1. 97
E.14 JSyn testresultaten van case 2. 98
E.15 JSyn testresultaten van case 3. 100
E.16 JSyn testresultaten van case 4. 101
E.17 JSyn testresultaten van case 5. 102
E.18 JSyn testresultaten van case 6. 103

1. Inleiding

1.1 Probleemstelling

Dit onderzoek vindt zijn oorsprong in de aloude concurrentie tussen analoge en digitale geluidsverwerking. Meer bepaald bij muzikanten, muzikaal artiesten en producenten die hun **live** opstelling uitrusten met allerlei analoge componenten. Denk hier aan effectpedalen, voorversterkers, equalizers, compressors, mengpanelen, *Direct Injection* boxen (DI-box) etc. Een productontwerper of IT-bedrijf zou al snel denken om hier digitale varianten van op de markt te brengen. Dat was inderdaad het geval voor veel mengtafelproducenten. Bekende merken zoals AKAI, Roland en Behringer zijn er al in geslaagd om volledig digitale mengtafels te produceren. Veel andere, ook kleinere bedrijven volgen hen. De digitale concurrenten hebben tal van voordelen op hun analoge voorgangers en wekken dus een grote vraag op in bij de consumenten.

Wat onveranderd gebleven is, is de aankoopprijs. Evenals hun analoge voorgangers kosten de digitale mengtafels al makkelijk 20.000€. Dit onderzoek wilt nog een stap verder gaan met de digitalisering. Deze paper probeert te achterhalen of zulke producten onder te brengen zijn in de vorm van puur software zonder bijkomende hardware. De aankoopprijs kan zo significant dalen en consumenten voeren alle operaties uit met enkel een standaard laptop.

1.2 Afbakening

De focus ligt puur op de vraag van de consument, de mogelijkheid tot intrede in de markt en de technische realiseerbaarheid.

1.2.1 In-scope

Het onderzoek beperkt zich tot een marktonderzoek en een technische test.

Om te zien of de overstap van analoog naar deze digitale omgeving realistisch is, wordt de technische test uitgevoerd. Daarin worden **real-life cases** uit de wereld van muziekproductie omgezet naar een **virtuele omgeving**.

De virtuele omgeving bestaat uit drie libraries voor geluidsverwerking. De libraries worden getest op performantie. Goede resultaten geven aan dat het technisch gezien realistisch is om de overstap naar digitaal te maken.

De real-life cases worden verkregen door middel van interviews in het marktonderzoek. Hier werden bands en producers gevraagd om de opstelling van hun live installaties uit te leggen. Deze installaties worden omgezet naar parameters voor de virtuele test cases. De parameters van de test cases zijn dus een discrete dataset en geen alomvertegenwoordigend spectrum van alle mogelijke installatiescenario's. Dit omdat het testen van een continue dataset teveel tijd in beslag zou nemen (na berekening blijkt 1275 dagen) en omdat meer dan de helft van de test cases redundant zouden zijn.

1.2.2 Out-of-scope

Het is duidelijk dat bij het onderbrengen van analoge producten in een software-pakket, de fysieke user-interface volledig wegvalt. Een fysieke interface is een groot argument pro analoog maar geen geldig argument tegen de potentie van digitalisatie. Daarom valt ergonomie buiten de scope van dit onderzoek.

1.3 Onderzoeksvragen

De hoofdvraag luidt als volgt:

Waarom blijft de muziekindustrie analoog werken in een digitale wereld?

Secundaire bronnen geven al snel een antwoord op deze vraag. Het antwoord is hetzelfde voor veel vakken die hun overstap van analoog op digitaal maken. Dat blijkt uit de artikels *Analog vs. Digital Synthesizers – My Take on the Old Debate* (**juliusdobos**) en *Analogue artists defying the digital age* (**GuardianOpinion**) alsook uit de interviews die voor dit onderzoek gevoerd werden. Daarin vonden we terug dat het grootste argument pro analoog de authentieke look-and-feel van de hardware is. Desondanks is dit geen tegenargument voor digitalisatie. Om hier dieper op in te gaan hebben we volgende deelvragen gesteld.

Is het mogelijk om digitaal te gaan?

Er zijn al tal van programma's waarmee men live aan geluidsverwerking kan doen. Deze programma's worden vaak gebruikt door DJ's en berusten altijd op externe harware. Deze deelvraag toont aan of de usecase van de DJ gegeneraliseerd kan worden naar andere actoren e.g. bands, producers, opname-artiesten etc.

Er wordt door middel van een literatuurstudie gezocht naar methodes voor digitale geluidsgeneratie en -verwerking. Van hoog belang is dat de methodes real-time toegepast kunnen worden door het doelpubliek; artiesten en producers.

Is het nuttig om digitaal te gaan?

Van alle mogelijke methodes voor digitale geluidsverwerking zal de scope vernauwd worden op slechts één methode. Deze methode wordt gekozen op basis van de mate waarin het intuïtief bruikbaar is voor en door het doelpubliek, de real-time capaciteit en de design capaciteit voor het modelleren van geluid. Zo beantwoordt deze vraag hoe de emotionele en empirische tests opgesteld zullen worden.

Is het realistisch om digitaal te gaan?

Op basis van de gekozen methode en hiervoor omschreven requirements worden drie libraries gezocht. Vervolgens worden er door middel van interviews naar real-life usecases gevraagd uit de muzieksector. En als laatste worden de specificaties van het teststation omschreven.

De real-life cases worden generiek omschreven in de libraries. De libraries zullen de cases uitvoeren en worden hierbij getest op performantie. Als er significant goede resultaten zijn, zelfs bij slechts één van de libraries, toont dat aan dat het technisch mogelijk is om digitaal te gaan. De empirische tests worden per usecase uitgevoerd. De resultaten van een test tonen enkel aan dat het realistisch is om digitaal te gaan voor die specifieke usecase.

Dit is de empirische test.

Is de overstap van analoog naar digitaal mogelijk?

Naast real-life usecases worden de geïnterviewden ook gevraagd in welke mate ze de digitalisatie verwelkomen en of ze geïnteresseerd zouden zijn in zulke innovaties.

Dit is de emotionele test.

Op basis van de resultaten van zowel de emotionele als empirische tests wordt per usecase bepaald of het mogelijk is om de overstap naar digitaal te maken.

1.4 Onderzoeksdoelstelling

De emotionele en empirische tests hebben elk twee uitkomsten: positief en negatief. Afhankelijk van de resultaten van de tests kunnen ondernemingen beslissen of het slim is om de markt te betreden met deze innovatie.

Emotionele test

- **Positief**: Uit de interviews blijkt dat de muzieksector geïnteresseerd is in digitalisering. Een onderneming kan succesvol deze markt betreden.
- **Negatief**: Deze innovatie wordt niet verwelkomd door de muzieksector. Wanneer een onderneming deze markt betreedt zal het geen vruchten afwerpen.

Empirische test

- **Positief**: Deze usecase is uitvoerbaar op het teststation. Het is technisch gezien mogelijk voor de actor om de overstap naar digitaal te maken.
- **Negatief**: Deze usecase is niet uitvoerbaar op het teststation. De actor van de usecase moet een sterkere machine hebben of blijft beter analoog te werk gaan.

1.5 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

2.1 Inhoud

In het hoofdstuk 1: Inleiding werd al gesproken over interviews, methodes voor geluidsgeneratie en libraries voor geluidsverwerking. In dit hoofdstuk zullen alledrie die onderwerpen verder uiteengezet worden. Deze onderwerpen vormen de primaire en een deel van de secundaire bronnen van dit onderzoek.

2.2 Interviews

Voor het opstellen van de empirische tests werd er gezocht naar real-life cases. Zo konden de tests representatief zijn voor usecases uit de muzieksector.

De cases zijn verkregen uit interviews met bands, artiesten en opnamestudio's. Deze sectie gaat over het verkrijgen van de interviews en de hoofdpunten ervan.

2.2.1 Correspondentie

Tabel B.1 toont alle artiesten die gevraagd zijn om geïnterviewd te worden. De selectie van mogelijke kandidaten hield rekening met volgende criteria:

- Er werden zoveel mogelijk instanties gevraagd zodat er van iedere branche[1] in de muzieksector zeker een interview verkregen is.

[1] Muzikant, artiest, band, opnamestudio, sound designer, producer, radio etc.

- Iedere branche kon live geïnterviewd worden zodat er mogelijkheid was tot doorvragen.
- De interviewee heeft ervaring met live optreden en kan spreken over zijn geluidsinstallatie en de good-practice ervan.

Alle correspondentie - de betrachtingen tot het verkrijgen van interviews - staan in tabel B.2. Iedere instantie heeft meerdere maals een verzoek gekregen.

Van de 25 gevraagde instanties heeft minder dan de helft geantwoord. Slechts bij vier van die helft is het gelukt om een interview af te nemen. Sommige instanties annuleerden hun afspraak. Andere instanties stuurden de aanvraag - soms cyclisch - door naar medewerkers.

De vier geslaagde interviews zijn opgenomen. U kan deze terugvinden in de bronverwijzing. Het doel van deze interviews was het verkrijgen van real-life usecases. Door de respons is de usecase van een branch verkregen van een enkele instantie. De omvang van de interviews is te klein om een representatieve steekproef te zijn. De usecases verkregen van deze instanties zijn wel gebruikt in de empirische tests. Dit door gebrek aan alternatieve data.

2.2.2 Bart Vincent (Producer)

Het opgenomen interview met Bart Vincent kan teruggevonden worden op GitHub (**bartvincent**). Er staat een transcriptie van dit interview in appendix C.1. Dit interview werd afgenomen op 11 april 2019.

Persona

Vincent is de producer voor artiesten zoals Sundahl, Kapitan Korsakov en de befaamde jazz zangeres Melanie De Biasio. Hij doet zowel de opname als het live geluid voor zijn artiesten en heeft ervaring met het schrijven en opvoeren van muziek.

Interview

Vincent meldt in zijn interview dat er reeds een evolutie gaande is in de muzieksector. Digitale mengtafels komen tegenwoordig veel meer voor dan twee à drie jaar geleden. Een must voor deze apparatuur is dat er een fysieke user-interface is (**bartvincent**).

In zijn werkproces is hij een idealist. In opname zoekt hij samen met zijn artiesten naar het gewenste geluid. Hij probeert dit vervolgens vast te leggen met oog op zo min mogelijk aanpassing in post-productie. Hij benadrukt dat een goede band en communicatie met de artiesten hier essentieel is (**bartvincent**).

“Er is zeer weinig dat nog real-time aangepast wordt,” meldt Vincent over zijn ervaring als geluidstechnicus (**bartvincent**). “Als een muzikant te luid speelt bij de sound check, dan communiceer ik dat ook. Maar live, tijdens het concert, pas ik daar weinig van aan.”

Vincent verwelkomt de digitalisering. Hij is van idee dat een computer nooit het werk van een mens kan volbrengen. In de tools die de mens bijstaan in productie ziet hij wel een verbeterende toekomst. “Het verwantschap met het instrument gaat anders zijn, maar het is nog steeds een instrument dat je kan leren spelen,” aldus Vincent (**bartvincent**).

Opstelling

Vincent spreekt zowel over zijn live opstelling alsook over zijn opstelling in opname van Melanie De Biasio. Zo’n 50-tal tracks krijgen elk equalization en compressie toegepast. Op de instrumentale tracks past hij vijf effecten toe. Dit wordt op twintig tracks geschat (**bartvincent**).

2.2.3 Peter Boone (Producer en Muzikaal Artiest)

Het opgenomen interview met Peter Boone kan teruggevonden worden op GitHub (**peterboone**). Er staat een transcriptie van dit interview in appendix C.2. Dit interview werd afgenomen op 25 april 2019.

Persona

Peter Boone staat gekend als een *jack of all trades* in de muziek wereld. Naast zijn job als producer gaat hij ook door als muzikant en arrangeur. Hij is het meest gekend als keyboardist voor EBM band A Split-Second waar hij in de jaren ’80 tot ’90 mee tourde.

Interview

“Op het moment dat de digitalisatie begon, kon het gewoon niet digitaler,” zegt Boone, “met momenten vond ik het ambetant dat zelfs een microfoon nog analoog was.” (**peterboone**) De prestatie van digitale mengtafels vond hij altijd maar teleurstellend. Dit ligt volgens hem aan de resolutie van het geluid. De meeste mengtafels werken in 16 bit aan 44.1 kHz wat hij CD-kwaliteit en beschamend slecht noemt (**peterboone**).

Er zijn mengtafels die 88.2 kHz en zelfs 192 kHz aankunnen. Deze tools zijn mede door het grote prijs etiket niet veel verspreid. Boone legt uit dat dit door een standardisatie in geluid. In de context van live muziek looft hij digitale mengtafels omdat alles veel sneller moet gebeuren. Het publiek gaat daarbij ook niet struikelen over de 44.1 kHz. “Het werkt zeer professioneel,” aldus Boone (**peterboone**).

Wanneer het echter aankomt op het vastleggen van het geluid, moet alles zo lang mogelijk analoog blijven. Boone meldt: “eender wat je `[aan 44.1 kHz]` van subtiliteit in `[geluid]` wilt steken gaat verloren omdat de notatie niet goed genoeg is. Eens je daar naar 88.2 kHz gaat wordt een galm terug een ruimte inplaats van een effectje.” (**peterboone**)

Desondanks dat Boone deel uitmaakte van een Electronic Body Music (EBM) band, is hij wel volledig afgestapt van hardware synthesizers. Tegenwoordig gebruikt hij enkel

nog software synthesizers. Net zoals de plug-ins in programma's voor audiobewerking zijn de algoritmes van software synthesizers exacte kopieën van wat de analoge machines achter de schermen doen. In tegenstelling tot de plug-ins hoor je, volgens hem, geen hoorbaar verschil tussen hardware synthesizers en de hedendaagse software varianten ervan (**peterboone**).

Boone zijn beeld van de digitalisatie is gemengd. Hij is zelf een grote voorstander van analoog, de keuze om een overstap naar digitaal te maken is volgens hem zeer afhankelijk van het genre van de muziek. "... de tijd gaat vooruit *whether you like it or not*. Ik ben er ... van overtuigd dat je *the best of both worlds* moet gebruiken." (**peterboone**)

Opstelling

EBM bands zijn zeer rechtuit. Waar bij de opnames van bands vaak instrumenten meerdere maals opgenomen worden om een *wall of sound* te creëren, zijn EBM bands zeer rechtuit. Voor dat genre worden er zowel op band als digitaal zo'n 24 sporen gebruikt.

2.2.4 Thomas Houthave (Sound Designer)

Het opgenomen interview met Thomas Houthave kan teruggevonden worden op GitHub (**thomashouthave**). Er staat een transcriptie van dit interview in appendix C.3. Dit interview werd afgenomen op 11 april 2019.

Persona

Na zijn studies aan SAE Institute Amsterdam vond Houthave al snel werk bij productiehuizen zoals Videohouse en Studio Brussel. In 2015 stichtte hij Klankwerk op waar hij nu zelf als sound designer en engineer werkt. Hij werkt met klanten zoals Ted-X, Philips en Abu Dhabi. (**klankwerkbio**)

Interview

Volgens Houthave was er in de jaren '90 reeds een eerste digitale evolutie in de muziek-sector. Veel consumenten namen snel de stap naar digitaal eens het na enige tijd ook goedkoper werd. Houthave spreekt veel over opstellingen en dat die vandaag vaak hybride zijn. Hiermee bedoelt hij dat er bij opnames een analoog mengpaneel gebruikt wordt, maar dat de opslag en bewerking zeker digitaal zijn. (**thomashouthave**)

Omdat Houthave commercieel en op korte deadline met zijn klanten werkt, heeft hij veel baat bij digitaal. "Sommige mensen zweren bij analoog maar tegenwoordig staat het zo dicht dat alles subjectief is," aldus Houthave. (**thomashouthave**)

Hoewel zijn professionele werkomgeving - naast de opnameapparatuur - reeds volledig digitaal is, ziet Houthave meer potentieel in software dan wat er momenteel op de markt is. Voornamelijk met het gebruik van AI. Hij spreekt in zijn interview over de *Master*

Assistant van Izotope Osone 8, een programma voor geluidbewerking. Het is een feature die equalization instellingen kiest voor de gebruiker op basis van een vooraf gegeven referentie track. De assistant zal proberen om het geluid van opnames zo goed mogelijk te laten klinken als de referentie track. Houthave hoopt dat de toepassing van AI op zulke, alsook op andere features meer onderzocht kan worden. (**thomashouthave**)

Opstelling

Houthave werkt met zelfgemaakte of gekochte samples. De samples moeten weinig bewerkt worden; ze zijn al kant en klaar. Naast de standaard equalization en compressie worden er dus weinig tot geen effecten gebruikt.
Voor zijn projecten gebruikt Houthave ongeveer 100 tot 150 tracks. (**thomashouthave**)

2.2.5 Vagabundos (Band)

Het opgenomen interview met Vagabundos kan teruggevonden worden op GitHub (**vagabundos**). Er staat een transcriptie van dit interview in appendix C.4. Dit interview werd afgenomen op 17 april 2019.

Persona

Vagabundos is een Gentse funk band met enkele Britse en Braziliaanse roots. Ze zijn opgericht in 2014. Na de release van een album, twee EP's en meerdere internationale tours maken ze zich nu klaar om een nieuw album aan te kondigen. Het is een charismatische groep met een grote passie voor zowel hun werk als het creatieproces dat erachter zit. (**vagabio**)

Interview

Refereer naar tabel C.1 voor de deelnemers van dit interview.

Na vijf jaar op de planken en snel gegroeide faam heeft Vagabundos kennis gemaakt met reeds veel stijlen van productie. Ze zijn het meest bekend met de opnamestudio van Peter Boone. Zijn werkproces staat beschreven in sectie 2.2.3. Naast Boone bespreekt Vagabundos in hun interview ook nog het productieprocessen van BOMA studio in Gent en van hun eigen gitarist Saulo Soneghet. (**vagabundos**)

BOMA studio kon niet geïnterviewd worden voor dit onderzoek. Zo staat in tabel B.2.

Vagabundos bespreekt het productieproces van Soneghet. Soneghet wordt beschreven als een perfectionist in zijn werk. Hij hangt veel belang aan postproductie. De band vertelt over hoe Soneghet een nummer zodanig veel op kan blinken tot er bijna geen human touch meer in de herkennen is. In opname neemt hij alles apart op zodat de het brongeluid zo klaar mogelijk is. (**vagabundos**)

Volgens Vagabundos is er geen goed of slecht in opname. Iedere producer of band heeft een eigen manier. De manier is meest afhankelijk van het gewenste geluid. “Peter is Peter and he likes another kind of sound so he records it in another kind of way,” meldt Adam Wilson, frontman van de band. (**vagabundos**) Boone meldde in zijn interview al dat bepaalde genres meer baat zouden hebben bij de digitalisatie dan andere. (**peterboone**)

Vagabundos staat open voor een digitalisatie en zien er zeker de voordelen van in. Toch uiten ze hun gemis voor de fysieke user-interface en de authenticiteit van het materiaal. Een natuurlijke reactie voor muzikanten. (**vagabundos**)

Opstelling

Het aantal tracks dat Vagabundos gebruikt voor hun live opstellingen loopt al snel op tot 20. Dat worden er 40 in opname. Adam (Woodie Bundo) Vandenhaute meldt dat hij op zijn basgitaar uitzonderlijk maximum drie actieve effecten heeft. Meestal zijn dit er maar 1 of 2. (**vagabundos**)

Soneghet heeft ook ervaring met productie voor metal bands. Daar gebruikt hij 50 tracks met 5 effecten op de instrumentale tracks daarvan. (**vagabundos**)

Alle sporen krijgen equalization en compression. (**vagabundos**)

2.3 Methodes voor Geluidsgeneratie en -verwerking

methodes beschrijft in zijn artikel vijf fundamentele methodes voor geluidsgeneratie. Voor dit onderzoek werd gezocht naar een methode die voldoet aan volgende selectiecriteria.

- Welke methode de muzieksector het intuïtiefst kon gebruiken.
- Welke methode programmatorisch het efficientst werkt.
- Welke methode het meest design capaciteiten biedt aan artiesten en sound designers.

De vijf fundamentele methodes beschreven door **methodes** zijn de volgende.

- Subtractive Synthesis
- Additive Synthesis
- Granular Synthesis
- Wavelet en Corpus-based Synthesis

De werking van deze methodes en de selectiecriteria worden hieronder verder toegelicht.

2.3.1 Subtractive Synthesis

Bij subtractive synthesis start de gebruiker met een harmonisch rijke geluidsgolf. Een golf is harmonisch rijk wanneer het veel subfrequenties bevat. Voorbeelden hiervan zijn zaagtand-, blok- en driehoeksgolven Wanneer de golf puur gehoord wordt zal de funda-

mentele frequentie het dominantst klinken (**harmonics**). Het is de taak van de gebruiker om door middel van frequentiefilters de ongewenste delen van het frequentiespectrum weg te halen en zo het geluid te modelleren (**subtractive**)

subtractive bespreken in hun artikel methodes voor digitale geluidsgeneratie en -verwerking. Deze methodes zijn slechts een benadering van wat analoge synthesizers doen. Maar in tegenstelling tot andere pogingen om analoge geluidsgeneratie te digitaliseren zijn de resultaten van deze methodes niet te onderscheiden van het analoge geluid (**subtractive**).

Subtractive synthesis is tot vandaag nog steeds de standaard voor geluidsdesign sinds Robert A. Moog de eerste subtractive synthesizer modules uitbracht in de jaren '60 (**subtractive**). **guitarpedals** beschrijft in zijn artikel hoe sommige effecten, die vandaag terug te vinden zijn in de effectpedalen voor gitaristen, digitaal verwerkt kunnen worden; allemaal door middel van subtractive synthesis.

2.3.2 Additive Synthesis

Additive synthesis is de tegenhanger van subtractive synthesis. Sinusgolven staan er gekend om maar één frequentie te hebben; hun fundamentele frequentie. Bij additive synthesis gaat de gebruiker tal van sinusgolven bij elkaar optellen om complexere geluiden te bekomen (**additive**).

Voor kleine toepassingen is deze berekening nog haalbaar voor een digitale workstation. Maar van het moment dat de geluiden complexer en harmonisch rijker moeten zijn, loopt de doorlooptijd van de generatie linear op met het aantal sommen van de additieve golf (**additive**).

Wanneer men harmonisch rijke golven wilt genereren zoals de zaagtandgolf, is dit onmogelijk te volbrengen op additieve wijze door het oneindig aantal sommaties dat berekend moet worden (**harmonics**).

2.3.3 Granular Synthesis

Bij granular synthesis start de gebruiker met een of meerdere zeer korte geluidsfragmenten - grains - die aanzien worden als een periode van een geluidsgolf. De gebruiker zal die periode verschillende keren kopiëren en verschalen. De periode krijgt vervolgens ook een aanvangsttijd toegewezen. Door verschillende grains op die manier met elkaar te vermengen, kan de gebruiker het gewenste geluid modelleren (**granular**).

Granular synthesis is van eigen al een techniek die enkel digitaal toegepast kan worden. Met de IT-sector als doelpubliek is het een zeer aantrekkelijke methode voor sound synthesis. De techniek is zeer bruikbaar voor geluidssimulaties en -imitaties, vooral door middel van een AI (**granular**).

granular legt in zijn paper zelfs een manier uit om in real-time (live) aan granulair geluid te genereren. Zelf geeft hij aan het einde van zijn artikel wel toe dat hij geen toekomst ziet

voor granular synthesis in de muziekproductie. Dit omdat er teveel parameters zijn die de gebruiker moet beïnvloeden.

2.3.4 Wavelet en Corpus-based Synthesis

Deze twee generatie methodes zijn soorten van granular synthesis. Door hun verschil in implementatie zijn ze elk toepasbaar op andere vlakken.

Wavelet Synthesis

Wavelet synthesis biedt de gebruiker niet meerdere maar slechts één periode om het modelleerproces mee te starten. De periode, hier wavelet genoemd, is op voorhand specifiek gekozen door de gebruiker om het gewenste geluid te kunnen bekomen. Verder verloopt het modelleerproces volledig zoals granular synthesis (**wavelet**).

Deze methode wordt meer gebruikt wanneer men zeer gedetailleerd geluid wilt nabootsen. Het modeleerproces vereist dat de gebruiker een goed inzicht heeft in muziektheorie en vergt ook dat het geluid iteratief gemodeleerd wordt (**wavelet**). Real-time is het dus slecht toepasbaar.

Corpus-based Granular Synthesis

Deze granulaire generatiemethode haalt de grains op uit een databank - de corpus - van vooraf opgenomen geluiden. Deze methode wordt vooral gebruikt wanneer men een specifiek geluid wilt imiteren. Een zoekalgoritme of AI weet dan exact waar hij gelijkaardige grains terug kan vinden en probeert dan het originele geluid na te bootsen aan de hand van die grains (**methodes**).

Corpus-based is niet toepasbaar wanneer de gebruiker nieuwe geluiden wilt maken. Niet alleen heeft het dezelfde tekortkomingen als gewone granulaire geluidsgeneratie, het heeft ook nog extra vrijheidsgraden van welke grains te kiezen uit de hele corpus.

2.4 Libraries

2.4.1 JASS

jass ontwikkelden JASS (Java Audio Synthesis System) voor hun onderzoek naar efficiente geluidsgeneratie. Deze library synthetiseert zeven voorafgaande onderzoeken naar modellen voor geluidsgeneratie die bedoeld zijn voor geluidseffecten in video games en simulaties.

jass leggen uit hoe verschillende libraries verschillende doeleinden, implementaties en kosten hebben. Hun doel met JASS is om een library te schrijven die, naast alle requirements die opgelijst staan in hun paper, ook een breed scala van mogelijke toepassingen

heeft. JASS is dus bedoeld als all-round library voor om het even welke sector (**jass**).

De Code

JASS kan teruggevonden worden op de website van van den Doel (**jasscode**).

In hun paper leggen **jass** de innerlijke werking van JASS uit. Hier valt op dat het een modulaire[2] library is. De ontwikkelaar kan zijn eigen modules ontwikkelen door over te erven van gegeven abstracte klassen en interfaces (**jass**).

Dit is ook hoe programma's voor geluidsbewerking en (modulaire) synthesizers opgebouwd zijn. Modulaire programma's zijn intuïtief voor gebruikers uit de muzieksector (**bartvincent**).

Een ontwikkelaar instantieert een geluidsgenererende subklasse. Deze klasse erft dus van de `Out` klasse. Bijgevolg moet deze klasse de `computeBuffer` methode implementeren. Deze methode kan twee dingen doen. Ofwel genereert het een buffer. Ofwel verwerkt het een buffer die de klasse verkrijgt via een `Source` attribuut. Alle klassen die erven van `In` en `InOut` hebben een `addSource` methode waarmee een `Source` toegevoegd kan worden (**jass**).

Wanneer een stramien van verwerkte buffers uiteindelijk via een `SourcePlayer` aangesloten wordt op de audio output van de machine, kan de buffer gehoord worden. Evenals subklassen van `In` en `InOut` kan een `SourcePlayer` meerdere `Source`'s hebben (**jass**).

2.4.2 Beads

Beads is een open-source Java library gemaakt door Oliver Bown. Het project is in 2008 tot stand gekomen met behulp van Monash University in Melbourne en heeft in 2014 zijn laatste update gekregen (**beads**).

De library is bedoeld voor het schrijven van real-time[3] programma's voor geluidsverwerking. Java was hier onmiddellijk de taal van voorkeur omdat het open-source is. De ontwikkelaars melden op hun site dat ze een zeer flexibele IO-laag geïmplementeerd hebben. De library functioneert zo in verschillende contexten (**beads**).

Het doel van Beads is het vergemakkelijken audio-implementatie. Bown en zijn team hopen om de ontwikkeling van audio applicaties toegankelijker te maken voor de standaard programmeur (**beads2**).

[2] Iedere module heeft zijn eigen atomaire rol (oscillator, filter, compressor, versterker etc.). Door ze met elkaar te verbinden - of patchen - krijgt de gebruiker een zelf ontworpen stramien van parameters waarmee hij zijn geluid kan boetseren.

[3] Wanneer een library of programma multi-threaded werkt zodat de gebruiker met het geluid kan interageren terwijl het gegenereerd of bewerkt wordt.

De Code

Programmeren in Beads start met het instantiëren van een `AudioContext`. Deze klasse vraagt in zijn constructor naar een audio output device. Het selecteren van een audio output device is mogelijk in native Java (**beadsdocs**).

Beads biedt een paar reeds voorgeprogrammeerde oscillatoren aan. Deze zijn ondergebracht in de `Buffer` klasse. De golven kunnen afgespeeld worden met de `WavePlayer` klasse (**beadsdocs**).

Beads werkt, net zoals JASS, ook modulair. Klassen die erven van `UGen` implementeren de `addInput` methode. De methode vraagt een andere `UGen` als parameter wiens output de input wordt van de betreffende `UGen` (**beadsdocs**).

Om `UGen`'s te horen, moeten ze aangesloten worden op de `AudioContext`. De `AudioContext` kan meerdere `UGen`'s als input aanvaarden. Vervolgens moet de ontwikkelaar de `start` methode oproepen in `AudioContext` om het geluid af te laten spelen (**beadsdocs**).

2.4.3 JSyn

JSyn is - de naam verraadt het - een Java library die traditionele modellen van modulaire synthesizers imiteert (**jsyn**). Net zoals Beads is het real-time en open-source verkrijgbaar op GitHub (**jsyngit**).

Mobileer Inc. hebben naast JSyn ook twee andere software audio projecten gereleased: JMSL (Java Music Specification Language) en PortAudio. JMSL is een Java-based specificatie taal waarin gebruikers instrumenten en composities kunnen definiëren. PortAudio is een cross-platform audio IO-library voor C (**jsyn**).

De Code

Beschouw de documentatie van JSyn (**jsyndocs**). Daar vinden we een aantal herkenbare klassen terug zoals basis oscillatoren (sinus-, zaagtand-, driehoeks- en blokgolf), verschillende soorten filters (low-pass, high-pass, band-pass, multi-pole etc.) en zelfs effecten (delay, envelope etc.).

JSyn baseert zich op één moederklasse: `Synthesizer`. Wanneer een ontwikkelaar modules instantieert, moeten die via de `add` methode toegevoegd worden aan de `Synthesizer` vooraleer ze gehoord kunnen worden. De `Synthesizer` klasse staat in voor het starten van alle toegevoegde modules en beslist ook de geluidskwaliteit van de output (**jsyndocs**).

De genererende en verwerkende klassen hebben - net zoals in JASS - een methode die een buffer genereert of verwerkt. Kijk hiervoor terug naar 2.4.1. De methode heet `pullData` en verkrijgt de te verwerken buffers via de `UnitInputPort`-attributen van de klassen (**jsyndocs**).

JSyn is, evenals Beads en JASS ook modulair. Het verbinden van klassen is mogelijk

door de `UnitInputPort`- en `UnitOutputPort`-attributen van genererende en verwerkende klassen. Via de `connect` methode kan een ontwikkelaar een `UnitOutputPort` connecteren aan een `UnitInputPort` (**jsyndocs**).

Een stramien van verwerkte buffers, wordt via de `LineOut` klasse op de audio output van de machine aangesloten. Wanneer de ontwikkelaar de `start` methode oproept op zowel de `LineOut` als de `Synthesizer`, weerklinkt het geluid. Het `UnitInputPort`-attribuut van `LineOut` kan meerdere inputs ontvangen (**jsyndocs**).

Vergelijking van de Libraries

	Beads	**JASS**	**JSyn**
Real-time	Ja	Ja	Ja
Modulair	Ja	Ja	Ja
Buffer methode	calculateBuffer	computeBuffer	pullData
Synthesis methode	Subtractive	Subtractive	Subtractive

Tabel 2.1: Vergelijking van de drie testlibraries.

Wanneer we de documentatie van de drie libraries beter bekijken, valt op dat hun code in essentie hetzelfde doet. Niet alleen hebben ze alle gelijkenissen uit tabel 2.1, de architectuur van het verbinden van genererende en verwerkende buffers is duidelijk ook typerend aan sound libraries.

Uitzonderlijk bij JSyn is wel dat alle parameters een `UnitInputPort` zijn. Neem een low-pass filter (LPF) als voorbeeld. Gegeven een harmonisch rijke geluidsgolf gaat een LPF alle aanwezige frequenties hoger dan een gegeven cut-off frequentie weg filteren. Het te filteren geluid is hier de input - bij zowel JSyn als andere libraries is dit het geval. De cut-off frequentie is hier een parameter voor de module.
Bij de meeste libraries zijn zulke parameters statische getallen. JSyn behandelt zulke parameters als `UnitInputPort`'s (**jsyndocs**).

De filosofie van object-oriented programming is het modelleren van de echte wereld. In dit voorval is JSyn de beste representatie van echte modulaire synthesizers. Parameters van modules zijn maar zelden statische waarden. Artiesten willen een interactieve band hebben met hun geluid. Daarvoor moeten zulke parameters dynamisch aangepasbaar zijn (**vagabundos**). In het geval van modulaire synthesizers *"moduleert"* men parameters door middel van *"control voltage"* (**modular**).

3. Methodologie

3.1 Inhoud

Allereerst worden de **empirische tests** uitgevoerd. Deze tests gaan na of de digitalisatie mogelijk is op technisch vlak. Het doel van dit onderzoek is analoge apparatuur om te vormen in pure software zodat het toegankelijker is voor consumenten met een lager budget. Daarom werd voor de technische tests gebruik gemaakt van een standaard particulieren laptop. De code geschreven voor de uitvoering van de tests wordt samen met de specificaties van het testsysteem respectievelijk besproken in secties 3.2.2 en 3.2.1.

Zodat de testcases van de empirische tests representatief zouden zijn, werd er gezocht naar real-life usecases uit de muzieksector. Hiervoor zijn interviews afgenomen. De interviews zelf zijn besproken in sectie 2.2. In de interviews werden artiesten, producers etc. gevraagd naar hun opstellingen voor zowel in opname als voor live muziek. Aan hand van die informatie werden representatieve testcases opgesteld. De opstelling van de testcases staan uitgelegd in sectie 3.2.2.

In de interviews werd ook gevraagd of digitale varianten verwelkomd zouden worden op de markt. Dit is interessant voor software bedrijven die de muzieksector willen betreden. Als er wel degelijk vraag is naar zulke innovaties, weten zij dat het voordelig is om in die niche te gaan werken. Hoe de antwoorden van de interviewees verwerkt werden staat beschreven in sectie 3.3. Dit zijn de **emotionele tests**.

Component		
Type	**Naam**	**Kracht**
Processor	Intel Core i3-4010U	1.70GHz
RAM	DDR3L SDRAM	8,00 GB
Opslag	Hybride Drive	500 GB

Tabel 3.1: Specificaties van de Testmachine

3.2 Empirische tests

3.2.1 De Testmachine

Er werd gezocht naar een laptop met gemiddelde specificaties voor de hedendaagse markt. Als deze laptop de testcases al aankan dan geldt hetzelfde voor andere gemiddelde particuliere laptops en zeker voor workstations van opnamestudio's en backstage geluidstechnici.

De gekozen machine is een Lenovo Ideapad Flex-14. Het modelnummer is 20308. De specificaties van de machine staan in tabel 3.1. Op de machine draaide een verse installatie van Linux Mint Sylvia.

3.2.2 De Code

Keuze en Werking van de Libraries

Uit de interviews, besproken in sectie 2.2, bleek dat de muzieksector altijd met subtractive synthesis werkt bij hun geluidsgeneratie en -verwerking. Voor de empirische tests zijn dus drie subtractive sound synthesis libraries gekozen. Zoals vermeld in sectie 2.4 zijn dit Beads, JASS en JSyn.

Sectie 2.4.3 toont aan dat de drie libraries in essentie een gelijkaardige werking hebben. Iedere oscillator en geluidsverwerkende klasse voert een specifieke bewerking uit op een buffer. De buffers kunnen ook aan elkaar geconnecteerd worden; de output van de buffer wordt de input van een andere. Zo maakt de gebruiker een stramien van parameters die hij kan gebruiken om het geluid te modeleren.

Opstelling van de Testcases

De testcases zijn opgesteld aan de hand van de interviews. De interviewees werden gevraagd naar volgende specificaties van hun opstelling.

- Hoeveel audiosporen er gebruikt worden.
- Op hoeveel van die sporen equalization en compressie van toepassing is.
- Het aantal effecten dat op eenzelfde moment actief is op een spoor.
- Op hoeveel van die sporen het stramien van effecten actief staat.
- Hoeveel parameters ze op één moment moeten bedienen.

Interviewee	Profiel	Testcase index
-	Hobbyist	1
Vagabundos & Peter Boone	Live band	2
Vagabundos & Peter Boone	Band in opname	3
Saulo Soneghet	Metal band in opname	4
Bart Vincent	Jazz band in concert	5
Thomas Houthave	Sound designer	6

Tabel 3.2: Testcase index per profiel

Index	#Tracks	#EQ en compressie	#Effecten	#Tracks met effecten	#Buffers
1	4	4	0	0	4
2	10	10	2	5	40
3	25	25	2	10	95
4	40	40	3	15	165
5	50	50	5	20	250
6	150	150	0	0	450

Tabel 3.3: Testcase parameters per index

Het aantal sporen werd in code vertaald naar oscillatoren. Die oscillatoren werden geconnecteerd aan equalizers en compressors. Die output werd - indien van toepassing - ook geconnecteerd aan een stramien van effecten. Alle losse outputs van oscillatoren, compressors en effecten werden aangesloten op de audio output lijn van het programma. Op die manier konden de antwoorden weergegeven worden als volgende vier integere parameters.

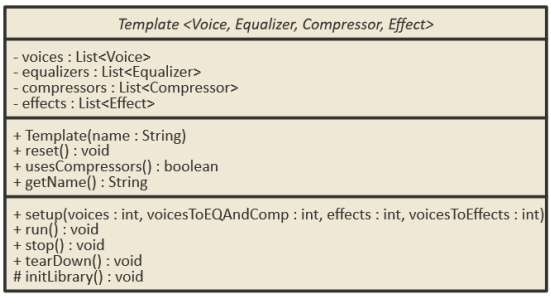
- Het aantal oscillatoren.
- Het aantal oscillatoren dat geconnecteerd wordt aan een equalizer en compressor.
- Het aantal effecten.
- Het aantal oscillatoren dat - eventueel na hun connectie aan een compressor - aan een stramien van effecten geconnecteerd wordt.

Voor het opstellen van de testcases is rekening gehouden met hoeveel buffers berekend worden in het programma. De testcases worden per library uitgevoerd in stijgende volgorde van het aantal gebruikte buffers. De paramerters staan per profiel weergegeven in tabellen 3.2 en 3.3.

Architectuur en Werking van de Test Framework

De code voor het uitvoeren van de testcases is terug te vinden in appendix D.

De volgende taak was om een test framework te maken. Met de parameters geformuleerd in tabel 3.3, moet het framework library-specifieke testcases uitvoeren en de performantie meten van het proces. Er werd rekening gehouden met uitbreidbaarheid zodat andere libraries makkelijk toegevoegd kunnen worden. Daarvoor werd de template pattern toegepast.



Figuur 3.1: Klasse diagram van de Template klasse

Figuur 3.1 toont het klasse diagram van de abstracte en generieke `Template` klasse van het framework. Dit is de moederklasse van de library specifieke testklasses. Wanneer een klasse zich uitbreid op `Template`, representeert de subklasse een sound synthesis library. De subklasse moet eerst de generieke types `Voice`, `Equalizer`, `Compressor` en `Effect` definiëren.

- **Voice** is een oscillatorklasse uit de library.
- **Equalizer** is een klasse uit de library die een inkomende buffer verwerkt. Om trouw te blijven aan de real-life testcases gebruikt men hier best een geluidsfilter. Geluidsfilters hebben karakteristieken van equalizers.
- **Compressor**: idem aan **Equalizer**.
- **Effect** is nog een verwerkende klasse. Deze klasse representeert een geluidseffect. Bart Vincent en Vagabundos gebruikten vaak een delay effect. (**bartvincent**) (**vagabundos**) Het wordt aangeraden om een gelijkaardig effect uit de library te kiezen.

In de subklasse worden vervolgens de `setup-`, `run-`, `stop-` en `tearDown`-methode overgeërft. Deze worden als volgt ingevuld.

- **Setup** vraagt de vier parameters uit tabel 3.3. Aan de hand van deze parameters worden de `Voice-`, `Equalizer-`, `Compressor-`, `Effect`-modules correct geïnstantieerd, geconnecteerd en opgeslagen in de desbetreffende attributen uit de `Template`-klasse.
- De **run**-methode start alle oscillatoren en verwerkende modules die in de `setup`-methode opgesteld zijn. Sound synthesis libraries starten in de meeste gevallen automatisch een nieuwe thread wanneer dit gebeurt. Als dit niet het geval is, moet de ontwikkelaar het zelf in een thread laten uitvoeren[1]. Dit is belangerijk voor de performantiemeting.
- De **stop**-methode stopt de thread - en bijgevolg ook het geluid - die in de `run`-methode opgeroepen is.
- **TearDown** haalt de opstelling van de testcase terug uit elkaar. Voor de modules verwijderd kunnen worden uit het geheugen, moeten ze eerst veilig gedeconnecteerd worden van zowel de library als van de andere componenten. Zo kan de library terug

[1]JASS start bijvoorbeeld geen nieuwe thread wanneer SourcePlayer.start() opgeroepen wordt. Om dit op te lossen werd de PlayThread-klasse ontworpen die de SourcePlayer eenmalig start tot de halt methode opgeroepen wordt.

```
top -b -n1 -d,01 | \\
grep %d | \\
awk '{ if (\$9 != \"0,0\") print \$9 \" \" \$10 }'
```

Figuur 3.2: Linux commando om performantie van een proces te meten.

volledig geïnitialiseerd worden voor de volgende testcase.

De uit te voeren testklassen en de parameters van de testcases worden statisch bijgehouden in de `StartUp`-klasse. In de `main`-methode wordt per testklasse iedere testcase uitgevoerd. Tijdens het runnen van de audio thread, wordt de performantie van het proces gemeten.

Performantiemeting van het Proces

Het voordeel van in een Linux-omgeving te werken is dat er geen externe library nodig is voor de performantiemeting. Alle nodige informatie kan al verkregen worden via het `top`-commando. De taak voor het afnemen van de metingen werd in het framework aan de `Measurer`-klasse gegeven. In die klasse kan commando 3.2 teruggevonden worden.

`Top` is een real-time commando. Wanneer het opgeroepen wordt in de terminal, worden alle processen en hun verbruik van CPU en geheugen dynamisch weergegeven. (**topcommand**) De volgende vlaggen worden toegepast.

- **-b** start het proces in *Batch mode*. Dit zorgt dat de header met algemene infromatie niet afgebeeld wordt zodat de output beter verwerkt kan worden door andere programma's of commando's.
- **-n1** maakt maar één meting in plaats van een continue dynamische output te geven.
- **-d,01** voert de meting op 0,01 seconden uit.

De `-n`-vlag wordt gebruikt omdat Java geen output van dynamische commando's kan lezen. De `-d`-vlag wordt gebruikt zodat de enkele meting zo snel mogelijk gemaakt wordt. Zo doende dook er een probleem op. Wanneer `top` standaard uitgevoerd wordt, update het proces ongeveer iedere de seconde. Wanneer men de metingen frequenter uit voert, wordt vastgesteld dat de meting voor `%CPU` onleesbaar is. Ongeveer iedere seconde wordt er een geldige meting afgebeeld. Daartussen vertoont de meting een `0` als resultaat.

Het interval van de seconde is niet betrouwbaar; soms was het meer, soms was het minder. Daarom wordt iedere 0,01 seconden een meting genomen. Ongeldige metingen worden genegeerd door het programma. Andere metingen worden opgeslaan tot er een totaal van 50 metingen per testcase is. Iedere meting duurt nog steeds ongeveer een seconde, maar van het moment dat een geldige meting verkrijgbaar is, wordt die ook opgenomen.

De output van het `top`-commando wordt doorgegeven aan het `grep`-commando. Java String formatting maakt gebruik van de `"%d"` om de *process ID* van het framework in te voeren in het commando. Zo verkrijgt men een enkele meting van het gezochte proces. `StartUp` heeft een statische methode `getPID()` die de *process ID* van het framework

$$y_i = med(x_{i-1}, x_i, x_{i+1})$$

Figuur 3.3: *Moving median* formule

$$h_i = (y_{i-1} + 2 * y_i + y_{i+1}) \div 4$$

Figuur 3.4: Hanning formule

teruggeeft. De output van `grep` wordt doorgegeven aan het `awk`-commando.

Het `awk`-commando filtert de `%CPU` en `%Mem` meting uit de output van `grep` op voorwaarde dat de `%CPU`-meting geldig is. De output van dit commando wordt ingelezen door Java. De twee metingen worden nog gescheiden door een spatie zodat het makkelijk te verwerken is.

3.2.3 Verwerking van de Testresultaten

Per library is er een dataset van 6 testcases. Iedere testcase heeft een subset van 50 metingen. Op zo'n subset wordt eerst een dubbele *median three* smoothing uitgevoerd. (**mediansmoothing**) Dit is een smoothing methode waarbij twee keer een nieuwe dataset gemaakt wordt. De nieuwe dataset wordt bekomen door een item te vervangen door de mediaan van zichzelf, zijn voorganger en zijn nakomer; zoals getoond in berekening 3.3. y_i is het item van de nieuwe dataset en x representeert de oude dataset. De eerste en laatste waarden worden overgenomen uit de oude dataset omdat berekening 3.3 niet mogelijk is op die indexen. Deze dubbele smoothing verwijdert korte periodes van extrema.

De resulterende dataset wordt vervolgens gladgestreken door middel van de Hanning vensterfunctie. Deze functie maakt dat plotse variaties in het frequentiedomein geëgaliseerd worden. Op die manier komen waarden die frequenter voorkomen in de dataset dominanter naar voor. Dit door het lopend gewogen gemiddelde van de waarden in de dataset te berekenen. De Hanning van een waarde in een dataset wordt berekend als de helft van die waarde opgeteld met een kwart van de voorgaande en nakomende meting. Dit staat afgebeeld in berekening 3.4. De eerste en laatste waarden van de nieuwe dataset worden overgenomen uit de oude dataset. (**hanning**)

Van de resulterende dataset wordt de mediaan genomen als resultaat voor de testcase. De mediaan is niet gevoelig aan extrema en geeft een betere representatie van de meer frequente metingen in een dataset. (**median**) Van de medianen wordt een trendlijn berekend. Zolang de richtingscoëfficient van de trendlijn van de testresultaten niet hoger ligt dan die van het aantal buffers in de input parameters uit tabel 3.3, kunnen we zeggen dat het technisch aanvaardbaar is om een overstap naar digitaal te maken.

3.3 Emotionele tests

Na de technische bespreking van de muzikale opstellingen van de interviewees, werden volgende vragen gesteld:

- Is er potentie voor digitalisering in de muziekindustrie?
- Wie zou hier het meeste baat bij hebben?
- Wie zou hier het meeste interesse in hebben?
- Zou een digitalisering verwelkomd worden in de sector?

Op basis van deze vragen, weten we of de branche van dat profiel in de muzieksector open staat voor een digitalisatie.

Er zijn door tijdsgebrek en de kleine respons (zie tabel B.2) slechts vier interviews afgenomen. Dit is duidelijk te weinig data om een populatie te representeren. Alle interviewees spreken uit jaren vakkundige ervaring. Daarom kan hun antwoord waardevol in aanmerking genomen worden.

Iedere interviewee beeld een profiel uit zoals in tabel 3.2 staat. Als resultaat op de emotionele test van een profiel wordt uit belang van dit onderzoek het resultaat van de interviewee zelf genomen.

4. Verloop van het Onderzoek

4.1 Is het mogelijk om digitaal te gaan?

Deze sectie gaat na welke methode van geluidsverwerking het best toegepast wordt bij het schrijven van verwerkingsprogramma's. Uit het interview van **thomashouthave** bleek dat hij het meest intuïtief overweg gaat met subtractive synthesis. In sectie 2.3.1 wordt besproken hoe deze methode werkt. **thomashouthave** meldt in zijn interview dat equalizers en compressors een uitwerking van subtractive synthesis zijn. De andere interviewees vermeldden ook dat zij extensief gebruik maken van equalizers en compressors. Het artikel van **filtervseq** biedt hier meer inzicht op.

4.1.1 Geluidsfilters

Filters maken prominent deel uit van de subtractive methode zoals in sectie 2.3.1 beschreven staat. **filtervseq** spreekt over de gelijkenissen en verschillen tussen audio equalization en filtering. Waar equalizers bepaalde delen van het frequentiespectrum van een geluid versterken, knippen filters ze af. Beide kunnen gebaseerd worden op een Fast Fourier Transformatie (FFT)[1]. Het enige verschil tussen de twee is de amplitudinale impact. Een equalizer versterkt het geluid rond een zekere frequentie terwijl een filter het verzacht.

Niet alleen biedt FFT meerdere toepassingen in subtractive synthesis. Het biedt ook een

[1] *fouriereq* leggen in hun onderzoek een toepassing van equalizers uit voor gehoorapparaten. Het idee is om een equalizer in gehoorapparaten te implementeren die bepaalde frequenties verluidt. Die frequenties worden gekozen op basis van het audiogram van de patient. Een audiogram geeft weer vanaf welke amplitude een patient een zekere frequentie kan horen. Zo worden enkel de moeilijk te horen frequenties versterkt. Dit wordt verwezenlijkt door middel van FFT.

```
Function<Double, Double> square = (x) -> (x % p) < (p / 2) ? -1.0 : 1.0;
```

Figuur 4.1: Java blokgolf generatie functie

tijdscomplexiteit van $\mathcal{O}(n \log n)$ die meer acceptabel is dan die van de Discrete Fourier Transformatie (DFT) van $\mathcal{O}(n^2)$.(**ffttime**) Het is niet optimaal, daarom worden er reeds tal van onderzoeken gevoerd om de tijdscomplexiteit te verminderen in software aan de hand van multicore computing. (**robbievincke**)

De bespreking van FFT is ter illustratie dat equalizers gebaseerd zijn op subtractive synthesis. Er zijn tal van toepassingen van FFT in geluidsfilters maar in software is het typisch niet de verkozen transformatie. Dit omdat het beter gebruikt wordt voor de spectroscopie van frequenties van continue signalen. De sound synthesis libraries besproken in sectie 2.4 maken hier gebruik van de bilineaire tranformatie (**jsynbiquad**) omdat die beter toepasbaar is op de real-time verwerking van discrete signalen[2]. **rbj** toont in zijn artikel hoe verschillende filter types geïmplementeerd kunnen worden aan hand van deze transformatie.

4.1.2 Harmonisch rijke golven

Op eerste zicht lijkt het per definitie onmogelijk voor een computer om harmonisch rijke golven te genereren. **fourier** vertellen het verhaal van Joseph Fourier. In de 19de eeuw stelde hij de hypothese dat alle periodieke functies beschreven kunnen worden als een oneindige som van sinusfuncties - vandaar de benoeming *harmonisch rijk*. Twee jaar voor zijn dood, in 1828, werd dit bewezen door Johann Dirichlet, een wiskundige met wie Fourier correspondentie voerde. Het zijn deze periodieke golven waar subtractive synthesis zich op baseert. (**fourier**)

Het is computationeel onmogelijk om een oneindige som van sinusfuncties te genereren. Maar in se is dat ook niet nodig. Evenals analoge synthesizers genereren de libraries uit sectie 2.4 hun harmonisch rijke golven door middel van logica in plaats van goniometrie. Zo kan de generatie van een harmonisch rijke functie vaak ondergebracht worden in één bewerking. Een voorbeeld in Java: functie 4.1 genereert een blokgolf met periode `p` gegeven een abcis `x`.

De resultaten van deze discrete functies zijn slechts een benadering van die van de continue Fourier series. (**fourier**) De geluidsresolutie van vandaag is hoog genoeg dat dit voor het menselijk gehoor nauwelijks zou mogen verschillen. (**vagabundos**)

[2] **rbj** bespreekt in zijn artikel zijn algoritme voor de bilineaire transformatie.

4.2 In het nuttig om digitaal te gaan?

De interviewees hebben elk laten weten wat zij dachten van een digitalisatie. Deze sectie bespreekt aan de hand van hun antwoorden of er in de markt vraag naar is. Het aantal interviews is te klein om een populatie te representeren. Desondanks spreken de interviewees uit jaren van vakkundige ervaring. Daarom kunnen hun antwoorden toch als representabel geacht worden.

4.2.1 Is het mogelijk om digitaal te gaan?

Bart Vincent: •
Thomas Houthave:
Peter Boone:
Vagabundos:

4.2.2 Wie heeft baat bij een digitalisering?

4.2.3 Zou een digitalisering verwelkomd worden?

4.3 Is het realistisch om digitaal te gaan?

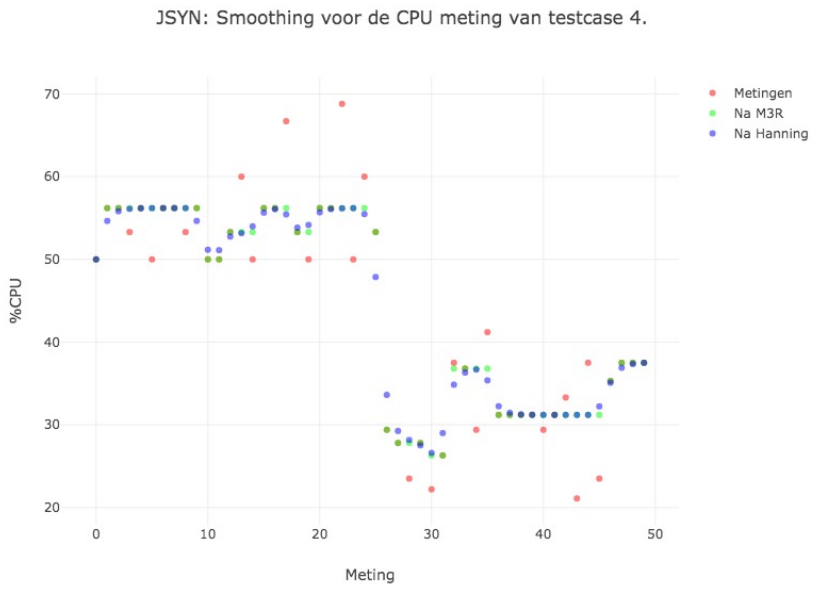
Voor dit onderzoek zijn er performantiemetingen uitgevoerd op de drie libraries die beschreven staan in sectie 2.4. In sectie 2.4.3 werd de gelijkenis in architectuur tussen de drie libraries vergeleken. Toch brachten ze verbazingwekkend verschillende testresultaten op. In deze sectie worden de verwerking van de resultaten besproken.

Om antwoord te kunnen geven op de onderzoeksvraag, zou slechts één van de libraries moeten slagen op deze test. Slagen voor deze test houdt in dat de library bij iedere testcase aanvaardbaar performant draait. Hou er rekening mee dat de tests op een ietwat zwakke testmachine draaiden die besproken is in sectie 3.2.1.

In sectie 3.2.2 wordt besproken hoe de testresultaten verkregen zijn. Deze zijn terug te vinden in appendix E. Vervolgens werden er twee smoothing methodes op de testresultaten uitgevoerd zoals besproken in sectie 3.2.3. In appendix F staan de resultaten van beide deze methodes. Hier wordt de verwerking van één testcase besproken: processor verbruik in testcase 4 van JSyn.

Figuur 4.2 toont de smoothing van een testcase. De rode punten zijn de originele metingen van de testcase. Als eerste wordt een dubbele *median three smoothing* (M3R) toegepast; besproken in 3.2.3. Dit resulteert in de groene set. M3R verwijdert plotse extrema uit het verloop van de meting. Vervolgens is op de resulterende dataset de Hanning vensterfunctie uitgevoerd. Dit leverde de blauwe puntenset op. Hanning maakt dat langere periodes van plotse extrema mooi geëgaliseerd worden.

De mediaan van de dataset die resulteerde uit de Hannign functie werd genomen als



Figuur 4.2: Smoothing van CPU testresultaten van testcase 4 voor JSyn

resultaat van de meting. In dit voorbeeld is de mediaan `48.93%`. Hieruit kunnen we concluderen dat JSyn doorgaans `48.93%` van de processorkracht gebruikt op de besproken testmachine.

Alle berekende medianen staan in tabel 4.1. Opvallend is dat JASS al snel aan op 100% processor vermogen zit. Waarom wordt duidelijk na het lezen van de broncode. Voor het opstellen van de testcases van JASS is de `Sine`-klasse gebruikt want het is de enige oscillator aanwezig in de library. **jasscode** vermeldt in de broncode van de klasse dat dit een *"highly inefficient implementation"* is.

Library	Case	Mediaan %CPU	Mediaan %Mem
Beads	1	6.23	0.70
	2	24.12	0.80
	3	68.80	0.80
	4	87.50	1.10
	5	92.59	1.50
	6	87.50	1.10
JASS	1	93.80	0.90
	2	93.80	1.0
	3	100.00	1.10
	4	100.00	1.20
	5	100.0	1.30
	6	100.00	1.90
JSyn	1	11.42	2.10
	2	19.05	2.30
	3	33.30	2.30
	4	48.93	2.30
	5	68.81	2.30
	6	56.20	2.2930

Tabel 4.1: Medianen van de verwerkte testresultaten per library per testcase.

5. Conclusie

5.1 Toekomstig Onderzoek

- Prototypering van een digitale mengtafel applicatie.
- Doe het niet in Java, doe het in C! Max...

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

Best-practice geluidsgeneratie en -verwerking
Onderzoeksvoorstel Bachelorproef

Victor Van Weyenberg[1]

Samenvatting
Audio hardware is duur. Software-based Digital Audio Workstations (DAW) bieden dankzij de groeiende werkkracht van persoonlijke computers een goed alternatief. Dit onderzoek levert via een vergelijkende studie tussen drie Software Sound Synthesizer (SWSS) libraries een best-practice voor geluidsgeneratie en -verwerking. Vereenvoudigde algoritmes creëren ruimte voor een betere geluidskwaliteit. Dit onderzoek zal een hulp zijn voor developers bij de keuze of creatie van hun SWSS's.

Sleutelwoorden
Onderzoeksdomein. Software — Development — Geluid — Generatie — Verwerking — Performantie — Efficiëntie — Kwaliteit

Co-promotor

Contact: [1] victor.vanweyenberg.w3751@student.hogent.be;

Inhoudsopgave

1 Introductie **1**
2 State-of-the-art **1**
3 Methodologie **2**
4 Verwachte resultaten **2**
5 Verwachte conclusies **2**
Referenties **2**

1. Introductie

Als tegenhanger van hardware synthesizers, maken software-based Digital Audio Workstations (DAW) de laatste jaren om tal van redenen steeds meer opgang. De werkkracht van persoonlijke computers groeide en groeit enorm. DAW's bieden een goedkoper alternatief op het aankopen van alle hardware die nodig is voor audiobewerking. Alle functies van de dedicated hardware kunnen ook ondergebracht worden in één DAW, dat maakt DAW's een meer polyvalent substituut.

Voor de development van DAW-software zijn er verschillende libraries beschikbaar bedoeld voor geluidsgeneratie en -verwerking. Deze libraries worden verder benoemd als Software Sound Synthesizers (SWSS). Dit onderzoek neemt de werking van drie SWSS's onder de loep. Er zal een vergelijking gemaakt worden tussen de broncode en performantie om zo een best-practice van software geluidsgeneratie en -verwerking te omschrijven.

Dit onderzoek zal een mapping creëren die voor developers de efficiëntie, performantie en kwaliteit van SWSS's en DAW's aantoont.

2. State-of-the-art

Pope (1993) heeft een gelijkaardige studie gedaan tussen de destijds drie meest gebruikte SWSS's. In zijn paper bespreekt hij *Music V*, een SWSS geschreven in 1967. Schrijven in *Music V* bestaat uit twee delen: de definitie van het instrument en het schrijven van de partituur. Voor het definiëren van instrumenten beschrijft *Music V* computer modellen van oscillators, filters, envelope generators, amplifiers etc. (Moog & Gamer, 2015) Door die modules correct te patchen, bekomt men het gewenste timbre. Naast het timbre kon men ook een partituur schrijven die instaat voor de melodie van de geprogrammeerde muziek.

Dit is een structuur die we ook in hedendaagse SWSS's terugvinden. *JSyn* (Burk, 2017), *Beads* (Bown, 2017) en *JMusic* (Brown, 2015) zijn drie open-source, Java-based SWSS's die aan die structuur voldoen. Omwille van de gelijkaardige structuur zijn deze libraries gekozen om vergeleken te worden in dit onderzoek.

Pope (1993) besprak *Music V* omwille van de simpliciteit. Dat was voor developers de grootste troef van de SWSS. Pope (1993) refereert naar *Music V* als de grondlegger van latere SWSS's. Om dit te illustreren bespreekt hij drie C-based SWSS's. Hij vervolgt met een vergelijkende studie op basis van architectuur en performantie.

De hierboven vermelde studie dateert van 1993. DAW's worden vandaag de dag nog uiterst zelden in pure C geschreven. Er is intussen een groot aanbod aan SWSS libraries. Dit onderzoek wilt de studie van Pope (1993) actualiseren gebruikmakend van de drie Java SWSS libraries.

3. Methodologie

De eertse stap is een vergelijking van de architectuur van de drie libraries met commentaar op vlak van efficiëntie.

De tweede stap bestaat uit een performantietest op geluidsgeneratie en -verwerking. Er worden verschillende testscenario's uitgeschreven met meerdere stemmen, bewerkingen en melodieën. Ieder scenario zal getest worden op dezelfde machine in ieder van de gekozen SWSS libraries. Hier wordt er ook rekening gehouden met de kwaliteit van de output.

Op basis van architectuur, performantie en kwaliteit wordt een best-practice omschreven voor software geluidsgeneratie en -verwerking.

4. Verwachte resultaten

Een oppervlakkige lezing van de broncode van de libraries, levert op dat JSyn bij het aanmaken van één oscillator al snel veel *UnitPort* objecten creëert. Deze *UnitPort* objecten worden gebruikt om de connecties met andere modules te maken en zo een stem te vormen. Omdat er al zoveel geheugen gealloceerd wordt per stem, wekt dit het vermoeden dat JSyn niet hoog zal scoren op de performantietest.

Een meer eenvoudige vorm van modules connecteren vinden we terug bij JMusic. Iedere opvolgende module in de zogenoemde *Chain* krijgt de voorgaande module mee als parameter in de constructor. Daarnaast valt ook op dat alle basis oscillator types ondergebracht zijn in één klasse. Dit met als nadeel dat de code van het genereren en verwerken van de audio buffer zeer groot wordt. JMusic lijkt in eerste opzicht niet de efficiëntste code te hebben maar scoort qua performantie wel beter dan JSyn.

Beads valt in het geluidsgeneratieproces al zeer snel terug op basis datastructuren en wiskunde. Voor ieder basis stem type is er - net zoals in JSyn - een aparte *Buffer*-klasse voorzien. Deze kunnen afgespeeld worden door de relatief kleine *WavePlayer*-klasse. Hypothese: Beads heeft de meest performante en efficiënte code.

De bespreking van code in dit hoofdstuk is zeer oppervlakkig en enkel beperkt tot geluidsgeneratie met als enige doel een hypothese op te stellen.

5. Verwachte conclusies

Hoewel JSyn niet de meest performante code heeft, bereikt hij met de extra geheugenallocatie een hogere geluidskwaliteit. De hiervoor besproken *UnitPort* objecten zijn bedoeld om modules te connecteren, wat ook de essentie van een SWSS is.

Wanneer men veel meerdere stemmen genereert of bewerkingen uitvoert, zullen er sneller kwaliteitsbeperkingen optreden bij JMusic en JSyn dan bij Beads.

Het KISS-principe levert de best-practice voor geluidsgeneratie en -verwerking. Het vereenvoudigen van algoritmes zoals geluid filteren of een blokgolf genereren bespaart geheugen zodat het programma meerdere stemmen kan genereren en bewerking kan uitvoeren zonder kwaliteitsverlies. De developer kan dan opteren om dat geheugen te gebruiken voor betere buffermanagement, betere verwerkingskwaliteit of hogere sample rates.

Referenties

Bown, O. (2017). *Beads JavaDocs*. Verkregen van http://www.beadsproject.net/doc/

Brown, A. (2015). *JMusic JavaDocs*. Verkregen van http://explodingart.com/jmusic/jmDocumentation/index.html

Burk, P. (2017). *JSyn JavaDocs*. Verkregen van http://www.softsynth.com/jsyn/docs/javadocs/

Moog, R. A. & Gamer, C. (2015). Electronic instrument. Verkregen van https://www.britannica.com/art/electronic-instrument

Pope, S. T. (1993). Machine Tongues XV: Three Packages for Software Sound Synthesis. *Computer Music Journal*, *17*(2), 23–54. Verkregen van http://www.jstor.org/stable/3680868

B. Correspondentie

Naam	**Functie**
Babek Joshghani	Dub artiest
Radio 2	Radio zender
Colin Benders	Electro artiest
UrgentFM	Radio zender
BOMA studio	Opname studio
Vagabundos	Band
JARVIN	Band
ROOM13	Opname studio
Peter Moorkens	Electro artiest
House of Media	Opname studio
Peter Van Praag	DJ
Muziekcafé Charlatan	Muziek café
Muziekcentrum Kinky Star	Muziek café
Hypestudio	Opname studio
Ben Van Camp	Muzikant
Muziekcentrum Goed Leven	Opname studio
Dante van Quaethem	Hoofdtechnicus Kinky Star
Frederik Sioen	Band
Mathias Sercu	Acteur / Muzikant
Bart Vincent	Producer
Thomas Van Elsander	Gitarist
Thomas Houthave	Geluidsingenieur
Frank Dûchene	Producer
Stijn	Artiest

Peter Boone	Opname artiest

Tabel B.1: Lijst van correspondenten en hun functies.

Naam	Datum	Medium	Beschrijving	Resultaat
Babek Joshghani	27/03/2019	Telefoon	Vraag voor interview	Overgaan op mail conversatie
JARVIN	28/03/2019	Telefoon	Vraag voor interview met Vagabundos	Doorverwijzing naar JARVIN
Babek Joshghani	28/03/2019	E-mail	Vraag voor interview	Interview vastgelegd
ROOM13	28/03/2019	Website	Vraag voor interview	
Colin Benders	28/03/2019	E-mail	Vraag voor interview	
Radio 2	28/03/2019	E-mail	Vraag voor interview	
UrgentFM	28/03/2019	E-mail	Vraag voor interview	
BOMA studio	28/03/2019	E-mail	Vraag voor interview	
Frederik Sioen	28/03/2019	Facebook	Vraag voor interview	
Radio 2	01/04/2019	E-mail	Extra uitleg geven over interview.	
JARVIN	02/04/2019	Telefoon	Tweede vraag voor interview	Interview vastgelegd
Babek Joshghani	02/04/2019	E-mail	Tweede vraag voor interview	In overleg
Ben Van Camp	06/04/2019	Telefoon	Vraag voor interview	In afwachting van antwoord
Peter Moorkens	06/04/2019	E-mail	Vraag voor interview	
House of Media	06/04/2019	E-mail	Tweede vraag voor interview	
Peter Van Praag	06/04/2019	Telefoon	Vraag voor interview	Interview vastgelegd
Muziekcentrum Goed Leven	06/04/2019	Telefoon	Vraag voor interview	Bericht achtergelaten op het antwoordapparaat
Muziekcentrum Kinky Star	06/04/2019	Telefoon	Vraag voor interview	Doorverwijzing naar hoofdtechnicus Dante van Quaethe
Dante van Quaethem	06/04/2019	E-mail	Vraag voor interview	
Hypestudio	06/04/2019	Website	Vraag voor interview	
Mathias Sercu	08/04/2019	Telefoon	Vraag voor interview, ook voor broers	Doorverwijzing naar Thomas VanElslander en Thomas Hou
Bart Vincent	08/04/2019	Telefoon	Vraag voor interview	Interview vastgelegd
Thomas Houthave	08/04/2019	Telefoon	Vraag voor interview	Interview vastgelegd
Thomas Van Elsander	08/04/2019	Telefoon	Vraag voor interview	Voicemail achtergelaten.
Ben Van Camp	08/04/2019	Telefoon	Vraag voor interview	Overgaan op e-mail correspondentie
Ben Van Camp	08/04/2019	E-mail	Vraag voor interview	

Stijn	09/04/2019	E-mail	Vraag voor interview		
Frank Dûchene	09/04/2019	E-mail	Vraag voor interview		
Peter Boone	17/04/2019	Facebook	Vraag voor interview	Interview op 24/04	
Peter Moorkens	24/04/2019	E-mail	Verdere planning interview	Geen antwoord	
Babek Joshghani	24/04/2019	Telefoon	Verdere planning interview	Geen antwoord	

Tabel B.2: Lijst van correspondentie met datum, medium, beschrijving en resultaat van het bericht.

C. Transcripties van de interviews

C.1 Bart Vincent (Producer)

Dus over de digitalisatie?
Dat is nu al bezig. Ik ben opgegroeid met analoge mengtafels, live. Er bestond toen een digitale tafel en die was *kut* eigenlijk. Maar die kon heel veel al op die momenten. Nu wilt niemand daar nog op werken omdat die pre-amps zo slecht zijn eigenlijk. Intussen is er zoveel geëvolueerd. In die tijd kreeg je een analoge mengtafel. Als er na u nog een band moest soundchecken moest je alles overschrijven. Van elke tafel kon je script sheets downloaden, daarop kon je aanduiden hoe de queues stonden. Intussen kom je bijna geen analoge tafels meer tegen. Laatste vier jaar ben ik live maar 4 analoge tafels tegengekomen. Hoe meer je de tafels leert kennen hoe meer je ermee overweg kan.

Mijn idee was dat alles nog analoog verliep.
Alles? Dat niet nee. Maar de stage blocks, micro's etc. Die verlopen nog altijd via een kabel, dat is analoog. Maar ze worden aangesloten op een digitale mengtafel.

Is die digitale mengtafel dan gewoon een computer?
Nee, het is een computer waarbij de schuivers een tool zijn die een computer aansturen. Na het selecteren van een track stel je de instellingen van die track in aan de hand van één input.

Al die stemmen worden allemaal verwerkt door de computer?
Ja, er verloopt altijd een analoog/digitaal conversie.

Klopt het dat, afhankelijk van de kwaliteit van de conversie, digitaal geluid slechter klinkt?

Nee, ik denk dat niet. Vroeger nam je op een band. Hoe breder uw band, hoe meer informatie je erop op kan slaan. De breedte kan je nu vergelijken met de 16- of 24-bit range van de geluidskwaliteit. Het is de hoeveelheid van 0'etjes en 1'tjes die je kan opslaan die maken hoe definieert uw geluid is. Ook de snelheid van de tape is belangrijk. Het aantal samples per seconde. En zeker belangrijk is de klok. Als je een accurate klok hebt, zal die de juiste samples op het juiste moment afspelen. Je kan soms enorm dure machines hebben die enkel dienen om een klok te genereren. Dat communiceert met uw DAW en dat zorgt dat uw soundbytes op het juiste moment gegenereerd worden.

Je werkt live; hoeveel stemmen gebruik je?
Ik werk op een beperkte versie van Protools, daar ben je gelimiteerd tot 128 tracks.

Gebruik je meer?
Nee, als je meer zou gebruiken, zou je al direct met een orkest samen werken. Als je een gewone pop band opneemt zou je genoeg hebben met zo'n 50-tal tracks.

Welke tracks zijn dat dan?
Een aantal voor doubling via een andere microfoon voor een andere kleurklank. Verder kan je als snel aan 12 kanalen komen met een drum kit.

Je bent een grote voorstander van zo min mogelijk postproductie te doen.
Ja. Iedereen moet aangevoerd worden. Als je een tom hebt die iet goed gestemd is, dan zit het in alle microfoons. Je kan sound replacement doen maar ik ben een voorstander van al te zoeken naar de klank die je wilt hebben in de studio, alvorens je gaat opnemen.

Ik zoek naar het real-time aanpassen van geluid. Wat doe je daar op een podium? Werk je constant met een heleboel parameters? Zijn dat er enkelen?
Alles wordt gesoundcheckt, het hele team wordt aangevoerd. Melanie Debiasio zingt van nature zeer stil. Als er dan een instrument te luid speelt dan communiceer ik dat ook. Als hij te luid speelt, komt dat in de micro van Melanie omdat die zo luid staat. Maar waar ik bijstuur, ik queue, ik compress. Met digitale mengtafels kan je alles al op voorhand instellen.

Hoeveel effecten gebruik je zo per track?
Ik zet er standaard 5 klaar.

Wat is jouw creatieproces?
Ruimte controleren, band en zangeres placeren. Maken dat iedereen zijn klank overeen komt hoe zij het willen. Dan is er een PA-repetitie; een generale repetitie met alle technici.

Is er een good-practice in jouw vak?
Er zijn een aantal regels. Natuurlijk kan je met digitaal niet in het rood gaan. Op een analoge tafel kan dat een leuk effect hebben.

En wat is het meeste werk tijdens het optreden zelf?
Ik luister. Soms stuur ik wat bij als het nodig is zodat alles overkomt zoals het hoort.

Is er ruimte voor automatisatie?

Onmogelijk, dat is allemaal afhankelijk van smaak. En iedereen heeft een andere smaak. Er is een plugin genaamd Izotope-8 die de gebruiker een mix voorstelt zodat het klinkt zoals een referentietrack. Maar dan kies jij, als gebruiker nog steeds of je het goed vindt of niet. Het hangt ook af van de ruimte. De charlatan gaat anders klinken dan het sportpaleis. Slates maakt een microfoon waarbij de karakteristieken van de microfoon digitaal kan instellen. Ik geloof het zelf niet want volgens mij werkt een groot membraan microfoon nog steeds met een groot membraan. Maar zo zijn er plugins, van verschillende merken, die zeker analoge pre-amps simuleren. En dat komt zeer dicht in de buurt. Ik heb verschillende pre-amps zowel analoog als digitaal gehoord en als je blind luisters zou ik niet kunnen zeggen welke hardware en welke software was.

Zijn er special technieken die eigen zijn aan jou?
Distortion om harmonieën bij te brengen.

Waarom hangen zoveel mensen vast aan analoog? Waarom maken ze de overstap naar digitaal niet ookal klinkt het hetzelfde?
Dat heeft te maken met smaak. Ik heb u gesproken over Peter Klaas. Ik ben bij hem op cursus geweest. Daar hoorden we twee keer hetzelfde nummer. éénmaal bewerkt met digitale gear; andere keer bewerkt met analoge gear. Een iemand kan vinden dat digitaal veel cleaner is. Anderen vinden dat ze karakter missen. Maar veel van dat "vuil"kan ook gegenereerd worden door een plugin. Het feit dat iets imperfect is maakt dat je soms tot iets onbedoelds maar origineels komt. Daar zoek je in productie soms wel naar.

Gebruik je soms andere manieren van geluidsgeneratie?
Nee.

[Ik geef een uitleg over verschillende soorten generatie.]
In welke methode past jouw vak het best?
Ik ben een man van compressie en equalizing. Mijn geluiden moeten passen in een geheel. Zo lijkt het het best op subtractive synthesis. Wat ik ook wel doe is veel doubling, het creëren van een "wall of sound". Doubling is twee keer hetzelfde proberen spelen. Het is onmogelijk om twee keer hetzelfde te spelen dus de kleine afwijkingen in het geluiden maken dat het veel vetter gaat klinken.

Is het mogelijk om digitaal te gaan?
Ik denk dat het nu al gebeurt. Dus ja! Zeker. In mijn leefwereld heeft alles te maken met de afstelling van de artiest. Dus of het allemaal goed gaat klinken, weet ik niet.

Wie zou er baat bij hebben?
Amateurs. Als je in een vlugge flow zit op radio, zou het zeker ook kunnen. Digitale gitaarpedalen bestaan al, Line Six is daarmee begonnen. Radio Head genereert zelf ook zijn eigen effecten. De verwantschap met het instrument gaat anders zijn, maar het gaat nogsteeds een instrument zijn waar je mee om kan leren gaan. Het fysieke gaat het meest gemist worden. Het is de look-and-feel die de muzikale ervaring versterkt.

Waarom zou digitalisering niet verwelkomd worden?
Het wordt wel verwelkomd maar een computer gaat gewoon nooit het werk van een mens kunnen overnemen. De tools, die de mensen helpen bij hun job, die worden wel beter.

C.2 Peter Boone (Producer en Muzikaal Artiest)

Hoe benoemt u uw functie in de muzieksector?
Muzikant, arrangeur, producer. Maar voornamelijk auteur componist. Het deel productie is ondergeschikt maar noodzakelijk. Ik wou niet afhankelijk zijn dus heb ik het mezelf aangeleerd.

Wat voor projecten heeft u zoal?
Het meest bekende is Split Second. Een Electronic Body Music Group (EBM) die in de jaren '80 mondiaal succes gehad heeft. We hebben wereldwijd in de top 10 gestaan en een LP of duizend verkocht in Amerika.

U heeft een uitgesproken mening over de digitalisering. Wat is uw creatieproces?
Hangt af van de context. Als ik een song schrijf komt eerst de tekst. Dan probeer ik het te accompagneren door ofwel piano ofwel tekst. Soms ontstaat het ook uit een bass-riff of een loopje. En voor je het weet heb je een song. En dat komt zeer organisch tot stand. Ik heb ondertussen zowel de digitale als analoge werkwijze toegepast en die werkwijze is eigenlijk niet veranderd. Ik maak het zeer organisch tot ik tevreden ben. Wanneer het stuk volledig naar mijn hand staat, noem ik het een afgewerkt product. Soms heb ik tracks teveel maar dan moet ik er later uitknippen. Less is more in productie. Een arrangeur bouwt op in zijn productie en een producer die schaaft het dan af tot een afgewerkt product.

Op dit moment, bent u digitaal of analoog bezig?
Ik heb er verschillende fasen in. Op het moment dat de digitalisering begon kon het niet digitaler zijn. Ik heb momenten gehad dat ik het ambetant vond dat zelfs een microfoon nog analoog was. Omdat, zeker als je niet in de top budgetten zit ga je sneller een goed resultaat behalen met digitale apparatuur dan met analoge apparatuur. Maar ondertussen heb ik geïnvesteerd in van alles en nog wat en heb ik gemerkt dat digitale mengtafels het eerste is dat ik heel zwak vindt. Vreselijk ontgoochelend van klank. Dat is het eerste dat ik uit mijn opstelling gezwierd heb. Maar dan moet je wel geld bovenhalen. Goedkope analoge mengtafels trekken op niks, hebben ook geen equalizers. Maar in de prijsklasse van 10-20 duizend euro kom je wel bij analoge mengers die beter zijn alles wat je op de markt vindt. Er is een heel vreemde stagnatie in digitale mengers. Zelfs de heel dure modelllen van sound craft werken nog altijd op een sample frequentie van 44.1 Hz en dat vind ik gewoon niet goed. 44.1 is CD niveau en de klank van een CD is beschamend slecht, nauwelijks beter van een cassetje.

Er is daar dus een grote tekortkoming?
Ja, absoluut, ik zit nog altijd te wachten op het volgende formaat, namelijk 88.2 en 24-bit. De CD is 44.1 en 16-bit. Eender wat je daar van subtiliteit in wilt steken gaat verloren omdat de notatie niet goed genoeg is. Eens je daar naar 88.2 gaat, wordt een galm terug een ruimte in plaats van een effectje. en als je analoog blijft, heb je dat probleem natuurlijk niet.

Waarom zou het moeilijk zijn om over te stappen naar 88.2?
Ik vraag het me ook af. Ik weet het niet. Maar zelfs de echt dure mengers zoals de VI-4 van Sound Craft die toch de standaard digitale menger aan het worden is in het digitale milieu

is 44.1 Hz. Pas op, Yamaha biedt wel menger aan die 88.2 en zelf 192 Hz aankunnen. Maar die zijn lang niet zo verspreid als andere mengers.

Zou dat liggen aan een standaardisatie?
Ja, natuurlijk, dat zal het wel zijn. Er is live en er is studio, dat zijn twee verschillende dingen. Als je mij live zou vragen wat ik vind van digitalisatie, dan noem ik het perfect. Daar zijn er geen analoge mengers meer. Alles werkt veel sneller, je kan ook zekere settings automatisch oproepen. Dat werkt ongelooflijk professioneel. Maar in de studio heb je wel tijd, en daar klinkt het beter als je het zo lang mogelijk analoog werkt en pas op het laatste moment overstapt op digitaal.

Leg eens verder uit hoe de werkwijze van analoog naar digitaal producen niet veranderd is.
Dezelfde regels gelden voor beiden. Als je met plug-ins knoeiwerk probeert op te lossen, ga je ook met knoeiwerk eindigen. Dat is zowel met analoog als digitaal zo. Ik zie dat met veel jonge artiesten. Ze bombarderen hun track met plug-ins waardoor je iets onnatuurlijks krijgt. Voor bepaalde genres - industriële popmuziek - is dat goed, maar voor ëchte muziekïs dat flauwe zever. Het klinkt niet goed. Is mijn mening. Met elektronische muziek kan ik mij voorstellen dat je geen boodschap hebt aan analoge toestellen. Hoewel sommige analoge compressors er wel een heerlijk effect op hebben. Voor je een zo'n digitale plug-in gebruikt, moet je hem hevig uitgetest hebben en weten waar je mee bezig bent.

Waarom is dat zo?
Die plug-ins zijn kopieën van analoge toestellen.

Wat zijn jouw maatregelen voor zo min mogelijk postproductie? Opstelling in de ruimte?
De klank van de ruimte moet goed zijn. In een kubusvormige ruimte krijg je staande golven en die moet je wegwerken door bijvoorbeeld blokken in de ruimte te plaatsen en de ruimte zo ongelijkmatig mogelijk te maken. Zo vermijd je dat, dat blijft zo in zowel het analoge als digitale milieu.

Werken jullie bij opname monofonisch of polyfonisch?
Dat ligt niet vast. Er zijn ook geen bindende voorwaarden. Sommige dingen klinken zelfs leuker als je ze apart trackt omdat je dan kan leren. Maar het heeft wel iets als je bepaalde instrumenten samen opneemt. Wanneer je dat doet én de instrumenten staan in dezelfde ruimte (zeer belangrijk), dan krijg je een ëmpathisch effect". Ze gaan beter op elkaar ingespeeld zijn. Je krijgt een soort van interactie en plezier dat je bij monofonie niet krijgt.

Is er een good-practice voor geluidsopname?
Close-mic alles. Maak dat je zeer goed gesoundcheckt hebt. Maak dat uw microfoons niet (half) in tegenfase staan. Zeker bij digitalisatie kunnen fases zelfs verschuiven. Maar dit is werk voor kenners en mensen die goede oren hebben. Alles is afhankelijk van de ruimte. Zeker bij drum. In een stenen ruimte kan je een drum soms opnemen met drie microfoons. Maar als je het veilig wilt spelen zet je overal beter nog een close mic bij.

Wat zijn de grootste verschillen tussen live en in opname?
Live zit je met een gitarist die naast de drummer staat. Als jouw gitaar te luid staat, krijg je

meer gitaar in de overheads dan drum. Bass, live, dringt overal in elke microfoon door. Dan moet je zeer hard opletten hoe je die positioneert. Live zit je vooral met overspraak problemen.

Doe je ook geluidsgeneratie?
Tuurlijk! Split Second!

Welke generatiemethode gebruik je dan?
Subtractive.

Gebruik je ook andere vormen?
Ik gebruik eigenlijk enkel virtuele (!) synthesizers. Er zijn nu virtuele digitale versies van oude Moogs waar je echt al een kenner moet zijn vooraleer je daar een verschil tussen hoort.

Heb je voorbeelden van die virtuele synthesizers?
Moog-1. Ik sta er eigenlijk niet meer bij stil wat ik gebruik. Ik ken mijn omgeving gewoon. Maar ik werk het meest in Logic (programma). Mijn synthesizers zijn vooral digitaal. Hoewel ik veel bezig ben akoestische projecten blijven die instrumenten toch digitaal voor mij.

Je zei dat digitale mengpanelen teleurstellend klonken...
Pas op, in studio. Live zijn het goede hulpmiddelen. Dit is het grote verschil tussen live en in studio. Je gaat dat ook merken. Iedere zelfrespecterende studio gaat een analoge mengtafel hebben.

Live toch digitaal werken, waarom?
Omdat het sneller gaat. Een soundcheck die normaal drie uur duurt, neemt bij een digitale menger maar een half uurtje in beslag.

Waarom is dat zo?
Omdat je meer apparatuur hebt.

Bij digitaal zit het meer geïntegreerd?
Ja, bij analoog duurt het lang voor je iedere knob geconfigureerd hebt. Bij analoog gaat dat veel sneller, je hebt persoonlijke presets die je met één druk op de knop automatisch in kan stellen.

Verstel je tijdens opname zekere parameters?
Hangt af van wat ik opneem. Bij bass bijvoorbeeld let ik zeker op hoeveel compressie ik gebruik. Anders krijg ik teveel niveau verschillen. Dat is van vitaal belang ook terwijl je aan het spelen bent. Bij gitaar - daarentegen - compress ik niet te veel. Daar zorg ik dat ik een gemiddeld niveau heb. Dat is wel een voordeel van digitaal t.o.v. analoog. Als je bij analoog een te laag opnameniveau hebt en je trekt het op, dan trek je ook de ruis op. en digitaal heeft geen ruis.

Is die compressie de enige parameter?
Wel, ik probeer een geluid te maken waar ik me comfortabel bij voel. Dit doe ik door mijn

equalization en compressie zo af te stellen dat ik nooit peak. Bij analoog kan dat nog een aangename distortion geven. Maar digitale distortion is fataal. Misschien komt het niet peaken uit mijn analoge geschiedenis maar iedereen doet het.

Vertel nog eens over Split Second.
Peter Bonne was de aanvankelijke producer. Hij kon een dag over de klank van een snare gaan tot hij het goed had. Wij waren wel de eersten die peperdure samples hadden.

Hoeveel tracks hadden jullie in opname?
Zo'n 24-tal, maar dat was nog op band.

Hoeveel zouden dat er huidig zijn?
Dat blijft hetzelfde, 24. Alles wat meer is, wordt meestal een ander genre muziek. EBM is meestal zeer rechtuit, dus je hebt daar niet al te veel dubbelende tracks.

Hoe zit het met de automatisering van het instellen van zekere parameters.
Alles wordt geautomatiseerd. Alle instellingen worden opgeslagen en kunnen met een druk op de knop recalled worden. Maar het automatisch instellen van parameters zal nooit geautomatiseerd kunnen worden. Ik zal het ook nooit van een programma laten afhangen hoe mijn opname gebeurt. Een computer zou het kunnen doen. Maar dat noem ik voor dummies.

Wat is een baat bij digitalisatie?
Het is toegankelijk voor iedereen. Vroeger moest je een miljonair zijn en dat is niet meer nodig. Dat is dubbel, het is de teloorgang van de opnamestudio want iedereen die een goede computer en drie microfoons heeft is al vertrokken. Vroeger moest je geld hebben. Met mijn eerste investering van 40.000 euro had een basis 16-sporen studiootje. Zonder het succes van Split Second zou dat niet gelukt zijn.

Wie heeft daar het meest baat bij?
Iedereen.

Is de digitalisering niet welkom?
Men kan daar zeer gefrustreerd over doen maar de tijd gaat vooruit en het is er whether you like it or not. Ik ben er wel van overtuigd dat je the best of both worlds moet gebruiken. Sommige dingen klinken analoog veel beter. Dus als je de (financiële) mogelijkheid hebt om dingen analoog te doen zou ik zolang mogelijk in het analoge milieu bezig blijven en pas helemaal op het einde naar digitaal overgaan. Pas op, dat geldt voor het genre muziek waar ik aan bezig ben. Maar daarom niet noodzakelijk voor alle genres. Leve de evolutie.

C.3 Thomas Houthave (Sound Designer)

In de jaren '90 is er een digitale revolutie ontstaan. Daarvoor werd er op tape opgenomen en plots werd alles digitaal opgeslagen.
Daar is zeer snel overgeschakeld omdat het naar gelang van tijd ook goedkoper werd. De puur analoge mengtafels die zijn er nu nog maar die worden weinig gebruikt. De meeste

opstellingen zijn nu hybride. Er zijn ook digitale tafels.

Wat zijn de digitale tafels? zijn dat gewoon computers?
Nee, echt hybride opstellingen. Deels analoog en deels digitaal in de zin dat je settings digitaal kan opslaan maar dat het signaalpad wel nog analoog is. Pre-amps, EQ's zijn analoog maar settings zijn digitaal.

Kan dat toegepast worden op effectpedalen?
Delay pedaaltjes zullen grotendeels digitaal zijn. Het is geen tape-delay, heeft geen tap en is bijgevolg dus niet analoog. Maar van het moment dat er AD-conversie is spreek je sowieso van digitaal.

Als er elektronische componenten gebruikt worden, maar geen computeralgoritme, is het dan digitaal of analoog?
Goede vraag. Daar ga ik niet op antwoorden. AD/DA-conversie is een vage grens. Maar wanneer je het geluid in 1'tje en 0'etjes veranderd, ben je digitaal bezig.

Laat die AD-conversie uw geluid slecht klinken?
Daar heerst een grote discussie. Maar dat is niet de vraag. Alles heeft zijn functies en charmes. Sommige mensen zweren bij analoog maar tegenwoordig staat het zo dicht dat alles subjectief is. Ik vind dat de voor- en nadelen van analoog of digitaal niet in de klank zit maar in het tactiele. Bij een analoge - zelfs bij digitale tafels - heb je knopjes om aan te draaien. Maar je hebt dat ook als plugin. Het verschil zit hem in de omgang met het instrument. De user experience heeft ook impact op de keuzes die je maakt. Het kan dan analoog of digitaal zijn. Je kan synthesizers hebben, hetzelfde analoog of digitaal. De keuzes die je maakt voor sound design gaan anders zijn afhankelijk van hoe je ermee interageert. Het is een andere beleving, dat is het grootste verschil. Ik hoor in ieder geval geen verschil. Niks is beter of slechter. Ik doe alles digitaal en ik vind dat wel lekker klinken.

Past u soms andere soorten geluidsgeneratie toe?
Ja, ik werk met pre-amps die dateren uit de jaren '70 die ik super goed vind klinken. Je kan zo'n goede EQ's hebben als je wil, maar als de bron van uw geluid niet goed is, krijg je het nooit rechtgezet. Je moet beginnen met een goede bron. Dat is dus een goede muzikant op een goed instrument dat binnengenomen wordt met een goede microfoon op een goede pre-amp, dan heb je al een vol, dynamisch geluid dat je volledig naar uw hand kunt zetten. Maar als dat al niet goed is, dan krijg je het nooit mooi. In de opleiding zeggen ze: ijou can't polish a turd".

Hoe genereert u uw geluid?
In sound design zijn dat dingen dat ik zelf opneem, dingen die ik sample of aankoop. Whatever works. Ik werk commercieel en met strakke deadlines dus het moet ook binnen zekere normen vallen. Het is niet dat ik strikt werk volgens bepaalde criteria.

Wat is uw werkproces?
Hangt af van de opdracht. Als ik tijd over heb, trek ik er wel eens op uit met een recorder en wat microotjes. De samples die ik opneem steek ik in mijn library en ik gebruik een software programmaatje om die terug te vinden. Die gebruik ik als basis geluiden. Maar ik

koop ze ook. Daar vind ik ook vaak mijn weg mee. Het kan echt vanalles zijn.

U vindt de digitale geluidsverwerking dus al goed?
Ja, dat vind ik. Ik werk altijd in Protools. Dat is een industriestandaard. Werkt zeer goed. Veel mensen werken daar ook op dus als er sessies gedeeld moeten worden, kan dat gewoon door één universeel bestandje door te sturen. Ik werk ook vaak onder beeld. Nu ben ik bezig aan een kortfilm. Je krijgt de dialogen die op set opgenomen zijn. Eerst maak ik die schoon en knip ik het op. Stel dat ik uit dit interview een woord zou knippen, dan valt er een stilte. Het is dan mijn taak om die stilte ook op te vullen zodat het natuurlijk en consistent blijft. Daarna EQ je het en leg je er ambiances onder. Zo maak je scenes, dat is redelijk basic sound design. Een andere soort sound design is het versterken van de realiteit, meer drama.

Gebruik je andere soorten geluidsgeneratie?
Nee, dat is iets volledig anders, dat heeft hier niks mee te maken. Het is verwarrend dat je dat generatie noemt. Ik zou het bron geluiden noemen. Die gebruik ik ook, maar dat is wanneer je dingen groter wilt laten lijken dan de realiteit. Stel nu dat je een gesprek op straat natuurlijk wilt laten klinken en er passeert een wagen, dan zet je gewoon het geluid van een wagen eronder. Stel nu dat de video uit verschillende perspectieven gefilmd is dan steek je er drie lagen wagen in met nog een woosh en een filtersweep. Een auto die passeert in the fast and the furious gaat compleet anders klinken dan in aanrijding in Moscou. Granulaire synthesis is dan weer muzikaler van aard. Daar definieer je instrumenten.

Leg granulaire synthesis eens uit.
[Houthave legt subtractive en granulaire synthesis uit.] Ik kan dat gebruiken maar dat is voor wanneer ik muzikalere sound design doe.

Hoeveel tracks gebruikt u in een project?
Dat kan al snel oplopen tot 100 - 150 tracks zijn. Dat is niet per se op één moment.

Hoeveel effecten zet u daarop?
Het beetje peper en zout: EQ en compressie. Om bepaalde frequenties te benadrukken. Dat op ieder spoor.

U bent ook live bezig.
Ja, ooit in mijn opleiding, maar ik vond dat veel te stressy. Mijn set op TedX was op zich geen live geluid. Dat was iets zeer ingewikkelds. Het was auditieve en visuele branding dat aangestuurd werd door mensen op het evenement. We hadden een soundscape gemaakt en we hadden al die verschillende lagen en elementen in Ableton gestoken. We hadden het zodanig geprogrammeerd dat al die verschillende branding die we gemaakt hadden getriggerd en aangestuurd werden door de mensen die het evenement bezochten. Ze zeiden iets in een voice booth, dat werd omgezet naar digitale code en dat beïnvloedde de visuele branding die op de grote schermen in de zaal speelden en ook de soundscape die in de zaal speelde.

Ziet u dat al snel analoog gebeuren?
Nee. Onmogelijk. Dat is te advanced. Analoog zou al willen zeggen dat je met tapes moet gebruiken dus misschien lukt dat voor een muzikaal artiest. Maar het zou nooit zo ver

kunnen gaan. Ik zie het niet gebeuren. Met generatieve patches op synthesizers lukt het misschien ook, maar wie wilt daar in godsnaam nog aan beginnen.

Wat zijn uw veiligheidsmaatregelen voor een direct goede opname.
Ja, ik ken mijn studio en test ook altijd alle apparatuur. Dat is niet veel werk op voorhand. Als je weet dat je gaat werken met een band, dan gaat dat proces al iets gevoeliger zijn. Ik heb ook een backup voor alles dat mis kan lopen. Ik heb zeker twee micro's en pre-amps voor één stem die opgenomen moet worden.

Heb je ooit al vreemde of excentrieke manieren van opname toegepast.
Ja. Er was eens een campagne voor Lidl waar we groenten en fruit gesampeld hebben en Tomorrowland anthems mee nagemaakt hebben. Er werden videoclips van gemaakt en als je kon raden welk liedje het was kon je tickets voor Tomorrowland winnen. Dat was al super experimenteel want als je met een banaan op een meloen slaat, klinkt het al zeer goed als kick drum.

Doet u veel sound design in de studio zelf?
Ja, zoals ik al zei: de bron moet goed zijn. Het valt de hard op als je geluid te veel gaat manipuleren. En soms is dat goed, dat is ook een stijl an sich. In de hiphop scene zit al veel vocoder en auto tune sound. Dat is een effect dat bedoeld was voor iets anders. Maar daar hebben ze wel hun sound van gemaakt.

Hoe zit het met automatisering van de instellingen van het geluid?
Ik denk dat dat al bestaat. Er is software van Izotope. Een bedrijf dat zich zeer hard op nieuwe technologie mikt. De doen audio magic. Ze hebben een plugin dat een geluid analyseert. Het herkent het geluid (instrument) dat dan automatisch sommige presets gaat instellen op het geluid. Iets met AI, daar gaan we wel naartoe. Voor het maken en componeren van muziek gaat dat ook meer gebeuren. Ik denk niet dat robots op een dag leuke muziek gaan maken. Maar ik denk wel dat je als muzikant of producer tools gaat hebben die het laten lijken alsof je met iemand aan het jammen bent. Dat je bijvoorbeeld een akkoord speelt en dat het programma er variaties op gaat spelen. Google heeft al een heleboel van die tools. Dat vind ik wel interessant. Je hebt nog altijd de human touch maar de tools worden wel aangereikt door een computer. Alsof je een bionische muzikant hebt. Waar ziet u AI nog zoal opduiken? In game engines. Maar dat is niet mijn wereld. Ik heb al sound design voor games maar nog niet op het niveau dat ik wil. Daarvoor moet je echt meer een programmeur dan een sound designer zijn. De kunst van een game sound designer ligt bij hoe hij de engine kan gebruiken om zijn geluid te vormen. Audio in een game engine werkt volledig anders. dat is zeer reactief. Bij ons verloopt dat lineair. In een game engine stopt het niet. Het geluid hangt aan objecten en de game engine weet afhankelijk van de positionering van geluiden hoe het moet klinken.

Is dat ook handig voor elektronische muziek artiesten.
Nee, het is een medium. Maar als componist is het wel interessant om een compositie te maken die modulair is. In games heb je het vaak dat wanneer het spannender wordt dat er elementen aan de muziek toegevoegd worden. Maar niet om nieuwe muziek te maken.

Is de synthesis methode determinerend vor hoe het geluid gaat klinken?

Alles heeft zijn eigen functie in geluid. Als je subtractive wilt werken, moet je subtractive werken.

Klinkt digitaal slechter dan analoog?
Er bestaat heel veel snobisme in analoge muziek. Audio is altijd al zoiets geweest waar ongelooflijk veel mensen hun specialist in zijn terwijl heel weinig mensen er hun basis van snappen. Als ik naar een Hifi winkel ga krullen mijn tenen van de uitleg dat ik krijg van de verkopers. Het is ook hip om nu een modulaire synthesizer te maken. Er zijn een heleboel mensen die hun eigen specialist worden. Het is en blijft ook een instrument. Dat dient om muziek te maken, niet om de vergelijking te gaan maken met andere instrumenten.

Kunnen door digitalisering sommige tools in een tablet gepropt worden?
Dat is al zo. IRig IK multimedia doet dat. Er zijn zodanig goede distortion plugins, waarom zou je ze niet gebruiken? OXBOX doet nu ook al aan software plugins die je kan aansturen met je tablet.

Denkt u dat muzikanten dat ook aantrekkelijk zullen vinden?
Ja.

Zou u daar baat bij hebben?
Ja ik denk er wel over om het ook te gebruiken. Het verbreedt mijn mogelijkheden. Vroeger had ik een multi-effect pedaal waar je alle effecten ter wereld mee kon nabootsen. Als ik daar nu naar luister klinkt het maar slecht.

Verwelkomt de muzieksector de digitalisering?
Ja, eigenlijk wel. Het ene moet het ander niet uitsluiten. Ik luister wel naar veel muziek op Spotify maar als ik een album echt mooi vind, dan koop ik het ook op vinyl.

C.4 Vagabundos (Band)

[c] **Naam**	**Functie**
Adam (Woodie Bundo) Vandenhaute	Bassist of Vagabundos
Adam Wilson	Vocals of Vagabundos
Luna Boone	Saxophonist of Vagabundos
Saulo Soneghet	Guitarist of Vagabundos
Peter Boone	Producer of Vagabundos
Jasper Boeur	Frontman and Producer of JARVIN

Tabel C.1: Deelnemers van het interview en vernoemde namen.

Vagabundos is a band of six people?
Woodie: No 5 now.

Oh, what happened?

Adam: You don't want to know. It's a professional personal slip up.

You've been touring to Brasil, also to England. Right now you're recording a new album.
Luna: Not yet.
Woodie: We're getting ourselves ready for the summer
Adam: Getting our shit together.
Woodie: we recorded the single ages ago and we're going to release it as a new one.

What's your process in creating new music? How do you go from nothing to something?
Woodie: sometimes we jam on stage.
Luna: Sometimes it's random improv
Woodie: When a lick sounds really cool, the drummer picks up on it. We start jamming and making a structure.
Adam: Everybody jumps in on little things and sometimes they jump in and play by themselves while everybody is jamming. In the process there's a lot of talking. There's a bassline, the drumline, the turn around and the middle eight section that might sound really cool but you might want to add more. Than you might have Saulo and Adam talking to each other about what they can do to harmonize with each other. Luna and I will be sitting there and some times it all comes together.
Luna: It gives my space as well to try out arrangements. Sometimes he comes up with a riff for me, sometimes Saulo does.
Woodie: We don't stick to our instruments. We invite each other to try and play something. Last time someone came up with a sick bassline for a dub part and he just pulled the bass out of my hands. Everyone is equal and together in the process of making music.

And the context of the music? The lyrics?
Luna & Woodie: The lyrics are completely from Adam.
Adam: Yeah, it's me. I like how a lot of bands write their lyrics together. But I do need another singer to write with. It's hard to write with a guitarist or a bassist. They don't have the same vibe. When a drummer talks to a drummer they know what they're talking about.

We talked about this earlier. How there used to be one microphone in the room when recording...
Woodie: Yeah, there used to be just one microphone in the room and the faders were like the distance from the instrument to the microphone. So the drum and the amps for the bass go way in the back. Otherwise it's too loud for the mic. And the singer comes real close, saxophone a bit more distant. It someone needs to be louder you just move them closer to the mic. That's how you fade your music.

You told me how you use different rooms when recording your music...
Woodie: Yeah, we record drums and bass together. The drums are in the room together with the bass. But the amp of the bass is in another room, which we record. The bassist has the sound in his headphones. That's how the drum and bass can play together while being recorded separately.
Luna: In the meantime I play along just for the completeness of the song. So everyone can

play together.

What are the safety measures you take to make sure your sound is recorded well?
Luna: At one moment I was playing my sax while being surrounded by a shield so my mic wouldn't record any other sounds than my own.
Woodie: The amp is one measure. But other bands can do it differently. They can choose to do a live recording. All together in one room. That way you have a lot of overwriting but it also provides another timbre.
Luna: It also depends on the space itself. We recorded in this studio and also in BOMA studio.
Woodie: Yeah, everyone had a different room there. The singer was looking down on us.
Luna: There was a whole "thingäround the drum kit and a really high ceiling.
Woodie: There were no parallel walls. It was a whole crooked place.

What the difference between live and in recording studio?
Adam: Our guitarist went to school for sound production and recording. He has a very clean way of doing things. He records everything separately. He then edits everything afterwards and sometimes it's a fucking pain in the ass for him to edit. While the way Peter (Boone) would do it is he would record everything together.
Luna: He likes the authentic sound. While Saulo is more a person who really likes to edit.
Woodie: Sometimes when there's a bassline and some notes are not exactly on time, he cuts in it and tries to fix it. That way it sounds "good"but also too clean. That way, when we listen to it, we miss out on something. People know us from playing live. When we release a clean-cut album like that there's just too much difference.
Adam: In my personal opinion: it didn't sound like us. Saulo loves it cause he can hear his work. He can hear everything perfectly, he's very proud it. But personally I don't want to record like this again.
Luna: We also discussed this for the next album.
Adam: He won't be editing it, basically.
Luna: He wants us to practice better so there doesn't need to be any editing.
Adam: Whilst creating our EP we rehearsed the shit out of it. We played the same song over and over again. If you can record that it sounds perfect. Of course there's still a human touch to it so it's not going to be "perfect perfect". But it sounded like us.
Adam: It's funny actually because we had our EP release booked before we'd even finish recording... Well, we finished recording the songs, but we hadn't finished editing them. So the EP release was booked and Saulo was working as fuck. And all of a sudden he said he couldn't finish it. So we just had to chop everything and be really fast with it and release it like this. That's why it still had a bit of authentic sound. But he had two years, you know? So I asked him ïs it finished yet?"And he's like "No, but we have to rerecord that."
Luna: It kind of milked it out. The album is not something I listen to anymore. But now I feel like we're getting to the point where we have enough material, again. And at some point, when we have time, I want to do it again but differently and better.
Woodie: Well rehearsed but not to much.
Luna: Yeah, so the band you hear is the same as you hear on stage.
Adam: We had a bit of an issue at one point where the album was recorded but it was still in editing process. And then we were still playing those songs on stage. Normally you make the album, release the album and tour the album, you know? Well, we did it the other

way around. So we ended up touring the album for three years and then it came out.
Luna: That's also how we make music. We make music on stage so by the time we get to finish the song as it is, we've also played it several times, several versions of it.
Woodie: The song also starts breathing when we play it live and realize what was good or bad or when the crowd goes crazy on something, we also change it in the original song. The song develops.
Adam: And then we play the same songs from the album and they sound different.

Is there a best-practice to recording or live music?
Luna: It really depends on the sound you want. It's very subjective.
Adam: Peter is Peter and he likes another kind of sound so he records it in another kind of way. On a drum kit there's a really good trick I learned. The microphone faces the skin of the drum. When you put your ear really close to the drum and you just play it whilst moving your ear around, you hear this *woop* but on some places you hear too much of the *kah*. When you place the mic you want to place it right between the *woop* and the *kah*. You really hear the tone of the drum as well as the *kah* when you hit the skin. Saulo really likes both of those things but sometimes you might want more of the *woop* or of the *kah*.
Woodie: The placement of the mics is really important with drums.
Adam: Jasper has something special on his amplifier. He marked a little square of where to put the microphone to record that specific sound he wants where it's on the edge of the cone.
Woodie: It's very subjective. If you want to record an old guitar you place your mic differently then when you're recording something else.
Adam: These are small things we do...

[Saulo enters the room.]

Woodie: We're talking about recordings. So it's right up your street.

Welcome, nice to meet you.
We were just talking about the good-practice of recording and how you record songs. I assume there is a good-practice, a öne way to do it perfectly".
Saulo: In terms of recording there is no right formula. I think that each approach you take at recording suits best for a different kind of music or band. We are a band where our energy manifests itself best when playing live. We are known to have very energetic live performances. So our first attempt is to reproduce this live energy in recording. But that also requires a lot of infrastructure before that's possible.
For instance: recording everybody live can be tricky because it's a way more complex studio setting in terms of separating the sounds from each other. It's not as easy as just putting everybody in a room, placing microphones and push "rec". I think the approach we try to use is to rehearse the main bulk of it live and then we remove a few things or redub a few things that need to be isolated. We also improvise a lot as well. Most of the solos come out of jamming and improvising.

Are there any personal tricks you use?
Saulo: Well depends. I'm known to be a producer that edits a lot. I like a lot of my sounds

to be super clean and distinctive so I can design every single instrument in the mix. Every single hit, every single tom on the drums needs to sound great. I clean a lot of spillage. But this is not a thumb rule. Depending on the genre you record some spillage can sound good. So I first clean up, and when it's all cleaned up I start bringing some of the dirt back in. I do this by adding a trash microphone or so and add a bit of ambiance. And I'm still a believer, especially when recording drums, that when you listen to the drum, as a player or as a listener, you listen to that with two ears. So when you record a multitrack session, you'd have to place up to 16-20 microphones around the drum kit. Sometimes they mic every single cymbal, metal band can do that sometimes. They put a lot of microphones around the drum but you don't listen to that with 16 ears. So the microphone that captures the kick drum should only capture the kick drum.

Why would that be useful?
Saulo: Because it's an accurate sound. And the idea to microphone things is in my opinion to translate the sound you would hear with your ears. Not to enhance it or make it any better. Unless you want something specifically designed for that. But in the case of Vagabundos I try to be accurate to the sound of the drummer. If I hear the drum like that with my ears, I also want to hear it like that in recording.

How many tracks do you use when recording songs? Depends, we have recordings with a lot of layers. It can very anywhere between 15 to 40 tracks. It also depends on the producer and recording studio. Sometimes we add harmonies to some of the instruments. We don't add too much stuff in the studio that would make it sound different from our live performances.

So you don't have that many effects on the tracks then?
Saulo: No, we use the basics. There are various effects but there is a basis. To me the main secret for a good mix all lays in two things: compression and equalization. If that's well done, you have a good sound to start from. For the rest, the only effects I use are ambients which are reverb, a bit of delay and that's it. Not everything has that. Normally vocals and saxophone are always going to have some reverb, delay, compression and equalization and that's it. The guitar is the same. Sometimes, when it's a dirty guitar, you don't have to compress it that much.

So there's not that much effect layering?
Saulo: No. We certainly don't do walls of fucking plugins, or walls of lots of effects, no.

Live there will never be 40 tracks, I guess.
Saulo: No, that's true. Live there would be something between 8 - 10 tracks. With some layering you easily reach something between 16 and 20 tracks.
Luna: Depending on how good you mic the drums, no?
Woodie: In practice it's different. Sometimes we have two or three mics for the drums.
Saulo: It depends on the venue. In small venues your don't have to mic the drums because it's too loud already. But when you go to a festival they sometimes double up on some of the drum's microphones. Our thumb rule is not to meddle with the live engineer's choice. I don't meddle too much with the live recordings. I only meddle with the engineering when it comes to recording.

Saulo and Woodie, you play guitar and bass. I have seen you use a lot of pedals...
Woodie: Yeah, for me they go straight to the amp. My signal comes from the bass through the pedals to the amp and then out. The pedals only transform the wave i play.

How many effects do you use at one time?
Woodie: At one time max three. That's very rarely. For some weird sounds I mix a lot of effects. Mostly we use one or two.
Saulo: with my guitar I have quite a lot of effects. I'm very rarely going to use two effects simultaneously. I have compression that is constantly on. Mainly so I can et some sustain on the pick-ups of my guitar. I always set the attack of the compressor very high so the plucking of a string with the plectrum really comes out. We play a lot of funk and that really accompanies the genre and enhances the tone. I have some distortions. Some that vary from grunch to heavy rock. But I never use them at the same time.

How many parameters can / do you have to control at once in your sound?
Saulo: At once?! Live I control very little. Everything is pre-programmed. The only parameter that I control is on the delays where I have a tap tempo function and the repetitions will go according to the song. Next to that I don't change them a lot. Only when sound checking.
Woodie: I have one bass micro synth. There are loads of parameters. So when we jam I fiddle with it. But distortion and boost always stay the same depending on the venue. I always have to check how loud I can go with my booster. After that we don't touch it anymore. Otherwise the sound engineer is going to get pissed of.

Have you ever thought about automation?
Saulo: Yes, I have thought about automating it. There are moments where I use different combinations of effects in the same song. There are some really needy systems that are really old and really complex to set up. Recently somebody told me about a Voodoo Lab system where you can preset your layers and just press a button to select the next or the previous combination.

Is it digital?
Saulo: I don't know.
Woodie: No it's not. It's fully mechanical. It uses tubes and air compression.
Saulo: I highly recommend you looking at Voodoo Lab. Because sometimes you start with a clean song. Then, halfway through the song you need a distortion with a delay.
Woodie: Yeah, you have to deactivate some sounds and activate some others and then you start tap dancing on your paddle board.
Saulo: I bear those things in mind when I create the arrangement for the songs as well. So when it comes to rehearsals I also rehearse which pedals to push.
Woodie: I always put the pedals I use frequently together close to each other on the board. When I have to activate the them I can press both the buttons with one step.
Saulo: If I count the amount of switches on my board, I have 16. So you really need to think it through. The effects is never the purpose. There are a lot of guitarists who think that that when get a lot of effects on a big pedal board, they're going to sound great. But in some of my songs I have my compression running and just put my distortion on and that's it.

Woodie: We're looking for our sound as well. In JARVIN, the other band, there I use my pedal board a lot. But here, with Vagabundos, I don't even bring my pedal board with me.

Could this pedal board ever be replaced with just a tablet?
Saulo: It could. Although I would not like it because most of my pedals are analogue. But I've worked in digital and emulation mode. I don't see problems with it. Nowadays it's all made ïn the box". There are some things for which I never use any digital plugins. But there are occasions where I do use them. To me they sound good and to me the final product is most important. You do not need to be a purist. Some say there is a clear difference. Is there?! 99.9% of the population cannot hear the difference. I challenge the .1% and I assure you they are bullshitting.

What's your opinion on the digitalisation?
Woodie: For me it's stupid to say but I like the pushing of the pedals. I like the authenticity. With an app that all disappears.
Saulo: You can still put it all into an IPad and have a pedal to control that. That way you can have the feel, but all the effects are processed by computer.
Woodie: Then you only have to bring your tablet.
Saulo: That's what digital mood effects do too, I think. They try to look vintage but it's all digital shit inside.

You were talking about metal bands. How many tracks do they use?
Saulo: They can easily go over 40 - 50 tracks. Sometimes they can do 10 layers of just rhythm guitar. They call it walls of guitar. And every track gets a different equalization. But I also believe that metal people are today the most prone people and the most open minded people to use everything digital. They are.

D. Code van de Empirische Tests

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package main;

import jass.engine.SinkIsFullException;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Reznov
 * @param <Voice>
 * @param <Effect>
 */
public abstract class Template<Voice, Equalizer, Compressor, Effect> {

    private String name;

    protected List<Voice> voices = new ArrayList<>();
    protected List<Equalizer> equalizers = new ArrayList<>();
    protected List<Compressor> compressors = new ArrayList<>();
    protected List<Effect> effects = new ArrayList<>();

    public Template(String name) {
        this.name = name;
    }

    public void reset() {
        voices.clear();
        equalizers.clear();
        compressors.clear();
        effects.clear();
    }

    public boolean usesCompressors() {
        return !this.compressors.isEmpty();
    }

    public String getName() {
        return this.name;
    }

    public abstract void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects);

    public abstract void run();

    public abstract void stop();

    public abstract void tearDown();

    protected abstract void initLibrary();

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package main;

import java.io.IOException;
import java.lang.management.ManagementFactory;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.Set;
import java.util.concurrent.TimeUnit;

import beads.BeadsTemplate;
import io.MeasurementWriter;
import jass.JassTemplate;
import jsyn.JSynTemplate;
import measure.Measurement;
import measure.Measurer;
import measure.Parameter;
import net.beadsproject.beads.core.io.JavaSoundAudioIO;

/**
 * @author Reznov
 */
public class StartUp {

    private final static List<Parameter> parameters = new ArrayList<>();
    private final Scanner input = new Scanner(System.in);

    static {
        /*int voices, voicesToEQandComp, effects, voicesToEffects;
        for (voices = 4; voices <= 150; voices += 2) {
            for (voicesToEQandComp = voices / 2; voicesToEQandComp <= voices; voicesToEQandComp++) {
```

```
            for (effects = 0; effects <= 5; effects++) {
                if (effects > 0) {
                    for (voicesToEffects = 1; voicesToEffects <= voices / 2; voicesToEffects++) {
                        parameters.add(new Parameter(voices, voicesToEQandComp, effects, voicesToEffects));
                    }
                } else {
                    parameters.add(new Parameter(voices, voicesToEQandComp, effects, 0));
                }
            }
        }
    }*/
    parameters.add(new Parameter(4, 4, 0, 0));
    parameters.add(new Parameter(10, 10, 1, 5));
    parameters.add(new Parameter(25, 25, 1, 10));
    parameters.add(new Parameter(40, 40, 3, 15));
    parameters.add(new Parameter(50, 50, 5, 20));
    parameters.add(new Parameter(150, 150, 0, 0));
}

private static Template[] templates = new Template[]{
        new BeadsTemplate(),
        new JassTemplate(),
        new JSynTemplate(),
};

public static void main(String[] args) throws InterruptedException, IOException {
    // new StartUp();
    run(templates[2], parameters.get(5));
    // System.out.println(System.getProperty("java.library.path"));
    // parameters.stream().filter(parameter -> parameter.getVoices() == 4).forEach(System.out::println);
    // JavaSoundAudioIO.printMixerInfo();
    // System.out.println(parameters.size());
}

private static void run(Template template, Parameter parameter) {
    template.reset();
    template.setup(parameter.getVoices(), parameter.getVoicesToEQandComp(), parameter.getEffects(), parameter.getVoicesToEffects());
    template.run();
    System.out.println(getPID());
}

public StartUp() throws InterruptedException, IOException {
    for (Template template : templates) {
        int test = 0;
        MeasurementWriter.open(template.getName());
        for (Parameter parameter : parameters) {
            template.reset();
            template.setup(parameter.getVoices(), parameter.getVoicesToEQandComp(), parameter.getEffects(), parameter.getVoicesToEffects());
            System.out.println(String.format("(%d) %s: Running parameters %s...", getPID(), template.getName(), parameter));
            TimeUnit.MILLISECONDS.sleep(1000);
            template.run();
            Measurer.measureTop(parameters.indexOf(parameter), parameter);
            // Measurer.measureSigar(parameters.indexOf(parameter), parameter);
            template.stop();
            template.tearDown();
        }
        MeasurementWriter.flush();
    }
}

public static final int getPID() {
    return Integer.valueOf(ManagementFactory.getRuntimeMXBean().getName().split("@")[0]);
}
}
```

```java
package measure;

import io.MeasurementWriter;
import main.StartUp;
import org.hyperic.sigar.CpuPerc;
import org.hyperic.sigar.Mem;
import org.hyperic.sigar.Sigar;
import org.hyperic.sigar.SigarException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.concurrent.TimeUnit;

public class Measurer {

    private static final Sigar sigar = new Sigar();
    private static CpuPerc cpuMeasurer;
    private static Mem memMeasurer;

    private static final String BASH_COMMAND = "top -b -n1 | grep %d | awk '{ if ($9 != \"0,0\") print $9 \" \" $10 }'";

    private static double getCpuLoadPercentage() {
        if (cpuMeasurer == null) {
            try {
                cpuMeasurer = sigar.getCpuPerc();
            } catch (SigarException e) {
                e.printStackTrace();
            }
        }
        return cpuMeasurer.getCombined() * 100;
    }

    private static double getMemoryLoadPercentae() {
        if (memMeasurer == null) {
            try {
                memMeasurer = sigar.getMem();
            } catch (SigarException e) {
                e.printStackTrace();
            }
        }
        return memMeasurer.getUsedPercent();
    }

    public static void measureSigar(int ID, Parameter parameter) throws InterruptedException {
        for (int index = 0; index < 50; index++) {
            Measurement measurement = new Measurement(ID, index, parameter, getCpuLoadPercentage(), getMemoryLoadPercentae());
            MeasurementWriter.write(measurement);
            System.out.println(measurement);
            TimeUnit.MILLISECONDS.sleep(10);
        }
    }

    public static void measureTop(int ID, Parameter parameter) throws InterruptedException, IOException {
        int index = 0;
        String[] command = new String[] {"/bin/sh", "-c", String.format(BASH_COMMAND, StartUp.getPID())};
        Process process; BufferedReader reader; String output;
        Runtime runtime = Runtime.getRuntime();
        while (index < 50) {
            process = runtime.exec(command);
            reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            output = reader.readLine();
            reader.close();
            process.waitFor();
            double cpuLoad = 0;
            try {
                cpuLoad = Float.valueOf(output.replaceAll(",", ".").split(" ")[0]);
                if (cpuLoad > 0) {
                    double memLoad = Float.valueOf(output.replaceAll(",", ".").split(" ")[1]);
                    Measurement measurement = new Measurement(ID, index, parameter, cpuLoad, memLoad);
                    MeasurementWriter.write(measurement);
                    index++;
                }
            } catch (NullPointerException ex) {
                ex.printStackTrace();
            } catch (NumberFormatException ex) {
                ex.printStackTrace();
            }
        }
    }

}
```

```java
package measure;

public class Parameter extends Tuple {

    private int voices;
    private int voicesToEQandComp;
    private int effects;
    private int voicesToEffects;

    public Parameter(int voices, int voicesToEQandComp, int effects, int voicesToEffects) {
        super(voices, voicesToEQandComp, effects, voicesToEffects);
        this.voices = voices;
        this.voicesToEQandComp = voicesToEQandComp;
        this.effects = effects;
        this.voicesToEffects = voicesToEffects;
    }

    public int getVoices() {
        return voices;
    }

    public int getVoicesToEQandComp() {
        return voicesToEQandComp;
    }
```

```
    public int getEffects() {
        return effects;
    }

    public int getVoicesToEffects() {
        return voicesToEffects;
    }

}
```

```
package measure;

public class Tuple {

    private Number[] data;

    public Tuple(Number... data) {
        this.data = data;
    }

    @Override
    public String toString() {
        StringBuilder tupleWriter = new StringBuilder(data[0].toString());
        for (int i = 1; i < data.length; i++) {
            tupleWriter.append(",").append(data[i]);
        }
        return tupleWriter.toString();
    }

}
```

```
package measure;

public class Measurement extends Tuple {

    public Measurement (int ID, int index, Parameter parameter, double cpuLoad, double memLoad) {
        super(ID, index, parameter.getVoices(), parameter.getVoicesToEQandComp(), parameter.getEffects(), parameter.getVoicesToEffects(), cpuLoad, memLoad);
    }

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package beads;

import main.Template;
import net.beadsproject.beads.core.AudioContext;
import net.beadsproject.beads.core.UGen;
import net.beadsproject.beads.core.io.JavaSoundAudioIO;
import net.beadsproject.beads.data.Buffer;
import net.beadsproject.beads.ugens.WavePlayer;
import net.beadsproject.beads.ugens.Compressor;
import net.beadsproject.beads.ugens.LPRezFilter;
import net.beadsproject.beads.ugens.Reverb;

/**
 *
 * @author Reznov
 */
public class BeadsTemplate extends Template<WavePlayer, LPRezFilter, Compressor, Reverb> {

    private AudioContext audioCtx;

    public BeadsTemplate() {
        super("Beads");
    }

    @Override
    public void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects) {
        initLibrary();
        int i, j;
        for (i = 0; i < voices; i++) {
            this.voices.add(new WavePlayer(audioCtx, 440, Buffer.SAW));
        }
        for (i = 0; i < voicesToEQAndComp; i++) {
            this.equalizers.add(new LPRezFilter(audioCtx));
            this.compressors.add(new Compressor(audioCtx));

            this.equalizers.get(i).addInput(this.voices.get(i));
            this.compressors.get(i).addInput(this.equalizers.get(i));
        }
        for (i = 0; i < voicesToEffects; i++) {
            for (j = 0; j < effects; j++) {
                this.effects.add(new Reverb(audioCtx));
                UGen previousModule;

                if (j == 0) {
                    if (usesCompressors()) {
                        previousModule = this.compressors.get(i);
                    } else {
                        previousModule = this.voices.get(i);
                    }
                    this.effects.get(i).addInput(previousModule);
                } else {
                    this.effects.get(i * effects + j).addInput(this.effects.get(i * effects + j - 1));
                }
            }
            audioCtx.out.addInput(this.effects.get(i * effects + j - 1));
        }
        for (i = i; i < voicesToEQAndComp; i++) {
            audioCtx.out.addInput(this.compressors.get(i));
        }
        for (i = i; i < voices; i++) {
            audioCtx.out.addInput(this.voices.get(i));
        }
    }

    @Override
    public void run() {
        audioCtx.start();
    }

    @Override
```

```
    public void stop() {
        audioCtx.stop();
    }

    @Override
    public void tearDown() {
        voices.forEach(voice -> audioCtx.out.removeAllConnections(voice));
        equalizers.forEach(equalizer -> audioCtx.out.removeAllConnections(equalizer));
        compressors.forEach(compressor -> audioCtx.out.removeAllConnections(compressor));
        effects.forEach(effect -> audioCtx.out.removeAllConnections(effect));

        reset();

        System.gc();
    }

    @Override
    protected void initLibrary() {
        JavaSoundAudioIO aio = new JavaSoundAudioIO();
        aio.selectMixer(2);
        audioCtx = new AudioContext(aio);
    }

}
```

```java
package jass;

import jass.engine.SinkIsFullException;
import jass.engine.Source;
import jass.engine.ThreadMixer;
import jass.generators.Delay;
import jass.generators.Mixer;
import jass.generators.OnePoleLowPass;
import jass.generators.Sine;
import jass.render.SourcePlayer;
import main.Template;


public class JassTemplate extends Template<Sine, OnePoleLowPass, Mixer, Delay> {

	private SourcePlayer sourcePlayer;
	private PlayThread playThread;

	private static final int BUFFER_SIZE = 64;
	private static final int SAMPLE_RATE = 44100;

	private class PlayThread extends Thread {

		private boolean started, running;

		public void run() {
			if (!running) {
				running = true;
				started = false;
				while (running) {
					if (!started) {
						sourcePlayer.run();
						started = true;
					}
				}
			}
		}

		public void halt() {
			running = false;
		}
	}

	public JassTemplate() {
		super("JASS");
	}

	@Override
	public void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects) {
		initLibrary();
		try {
			int i, j;
			for (i = 0; i < voices; i++) {
				this.voices.add(new Sine(BUFFER_SIZE, SAMPLE_RATE));
			}
			for (i = 0; i < voicesToEQAndComp; i++) {
				this.equalizers.add(new OnePoleLowPass(BUFFER_SIZE));
				this.compressors.add(new Mixer(BUFFER_SIZE, 1));

				this.equalizers.get(i).addSource(this.voices.get(i));
				this.compressors.get(i).addSource(this.equalizers.get(i));

				this.compressors.get(i).setGain(0, 1.0f);
			}
			for (i = 0; i < voicesToEffects; i++) {
				for (j = 0; j < effects; j++) {
					this.effects.add(new Delay(BUFFER_SIZE));
					this.effects.get(i * effects + j).setRawDelay(0.5f);

					if (j == 0) {
						Source previousModule;
						if (this.usesCompressors()) {
							previousModule = this.compressors.get(i);
						} else {
							previousModule = this.voices.get(i);
						}
						this.effects.get(i * effects + j).addSource(previousModule);
					} else {
						this.effects.get(i * effects + j).addSource(this.effects.get(i * effects + j - 1));
					}
				}
				sourcePlayer.addSource(this.effects.get(i * effects + j - 1));
			}
			for (i = i; i < voicesToEQAndComp; i++) {
				sourcePlayer.addSource(this.compressors.get(i));
			}
			for (i = i; i < voices; i++) {
				sourcePlayer.addSource(this.voices.get(i));
			}
		} catch (SinkIsFullException ex) {
			ex.printStackTrace();
```

```
            }
        }

        @Override
        public void run() {
                playThread.start();
        }

        @Override
        public void stop() {
                sourcePlayer.stopPlaying();
                playThread.halt();
        }

        @Override
        public void tearDown() {

                reset();

                System.gc();
        }

        @Override
        protected void initLibrary() {
                sourcePlayer = new SourcePlayer(BUFFER_SIZE, BUFFER_SIZE, SAMPLE_RATE, "default [default]");
                sourcePlayer.setOutputChannelNum(2);
                playThread = new PlayThread();
        }
}
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jsyn;

import com.jsyn.JSyn;
import com.jsyn.Synthesizer;
import com.jsyn.unitgen.*;
import main.Template;

/**
 *
 * @author Reznov
 */
public class JSynTemplate extends Template<SawtoothOscillator, FilterLowPass, EnvelopeAttackDecay, InterpolatingDelay> {

    private Synthesizer synth;
    private LineOut lineOut;

    public JSynTemplate() {
        super("JSyn");
    }

    @Override
    public void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects) {
        initLibrary();
        int i, j;
        for (i = 0; i < voices; i++) {
            this.voices.add(new SawtoothOscillator());
            synth.add(this.voices.get(i));
        }
        for (i = 0; i < voicesToEQAndComp; i++) {
            this.equalizers.add(new FilterLowPass());
            this.compressors.add(new EnvelopeAttackDecay());

            synth.add(this.equalizers.get(i));
            synth.add(this.compressors.get(i));

            this.voices.get(i).output.connect(this.equalizers.get(i).input);
            this.equalizers.get(i).output.connect(this.compressors.get(i).input);
        }
        for (i = 0; i < voicesToEffects; i++) {
            for (j = 0; j < effects; j++) {
                this.effects.add(new InterpolatingDelay());
                this.effects.get(i * effects + j).allocate(2);
                synth.add(this.effects.get(i * effects + j));

                if (j == 0) {
                    if (usesCompressors()) {
                        this.compressors.get(i).output.connect(this.effects.get(i * effects + j).input);
                    } else {
                        this.voices.get(i).output.connect(this.effects.get(i * effects + j).input);
                    }
                } else {
                    this.effects.get(i * effects + j - 1).output.connect(this.effects.get(i * effects + j).input);
                }
            }
            this.effects.get(i * effects + j - 1).output.connect(0, lineOut.input, 0);
            this.effects.get(i * effects + j - 1).output.connect(0, lineOut.input, 1);
        }
        for (i = i; i < voicesToEQAndComp; i++) {
            this.compressors.get(i).output.connect(0, lineOut.input, 0);
            this.compressors.get(i).output.connect(0, lineOut.input, 1);
        }
        for (i = i; i < voices; i++) {
            this.voices.get(i).output.connect(0, lineOut.input, 0);
            this.voices.get(i).output.connect(0, lineOut.input, 1);
        }
    }

    @Override
    public void run() {
        synth.start();
        lineOut.start();
    }

    @Override
    public void stop() {
        lineOut.stop();
        synth.stop();
    }
```

```
    @Override
    public void tearDown() {
        lineOut.input.disconnectAll();
        this.voices.forEach(voice -> {
            voice.output.disconnectAll();
            synth.remove(voice);
        });
        this.equalizers.forEach(equalizer -> {
            equalizer.input.disconnectAll();
            equalizer.output.disconnectAll();
            synth.remove(equalizer);
        });
        this.compressors.forEach(compressor -> {
            compressor.input.disconnectAll();
            compressor.output.disconnectAll();
            synth.remove(compressor);
        });
        this.effects.forEach(effect -> {
            effect.input.disconnectAll();
            effect.output.disconnectAll();
            synth.remove(effect);
        });
        synth.remove(lineOut);

        reset();

        System.gc();
    }

    @Override
    protected void initLibrary() {
        this.synth = JSyn.createSynthesizer();
        this.synth.add(lineOut = new LineOut());
    }

}
```

```java
package io;

import measure.Measurement;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;

public class MeasurementWriter {

    private static final String CSV = ".csv";
    private static FileWriter writer;

    public static void open(String fileName) {
        try {
            writer = new FileWriter(String.format("%s%s", fileName, CSV));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void write(Measurement measurement) {
        try {
            writer.write(measurement.toString().endsWith("\n") ?
                    measurement.toString() :
                    new StringBuilder(measurement.toString()).append("\n").toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void flush() {
        try {
            writer.flush();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

E. Testresultaten

E.1 Testresultaten van Beads

Tabel E.1: Beads testresultaten van case 1.

Index	%CPU	%Mem
0	87.5	0.6999999881
1	105.5999984741	0.6999999881
2	175	0.6999999881
3	31.2000007629	0.6999999881
4	20	0.6999999881
5	12.5	0.6999999881
6	6.1999998093	0.6999999881
7	13.3000001907	0.6999999881
8	6.1999998093	0.6999999881
9	6.6999998093	0.6999999881
10	12.5	0.6999999881
11	6.1999998093	0.6999999881
12	6.1999998093	0.6999999881
13	6.1999998093	0.6999999881
14	12.5	0.6999999881
15	6.6999998093	0.6999999881
16	6.1999998093	0.6999999881
17	6.6999998093	0.6999999881
18	6.1999998093	0.6999999881
19	6.1999998093	0.6999999881
20	6.1999998093	0.6999999881

21	6.1999998093	0.6999999881
22	6.1999998093	0.6999999881
23	6.1999998093	0.6999999881
24	13.3000001907	0.6999999881
25	6.1999998093	0.8000000119
26	6.6999998093	0.8000000119
27	6.1999998093	0.8000000119
28	6.6999998093	0.8000000119
29	6.1999998093	0.8000000119
30	6.1999998093	0.8000000119
31	6.6999998093	0.8000000119
32	6.1999998093	0.8000000119
33	12.5	0.8000000119
34	6.6999998093	0.8000000119
35	6.1999998093	0.8000000119
36	6.6999998093	0.8000000119
37	6.1999998093	0.8000000119
38	6.1999998093	0.8000000119
39	6.1999998093	0.8000000119
40	6.1999998093	0.8000000119
41	6.1999998093	0.8000000119
42	6.1999998093	0.8000000119
43	6.6999998093	0.8000000119
44	6.1999998093	0.8000000119
45	6.1999998093	0.6999999881
46	6.6999998093	0.6999999881
47	6.1999998093	0.6999999881
48	6.6999998093	0.6999999881
49	6.1999998093	0.6999999881

Tabel E.2: Beads testresultaten van case 2.

Index	**%CPU**	**%Mem**
0	156.1999969482	0.6999999881
1	187.5	0.6999999881
2	211.8000030518	0.8000000119
3	76.5	0.8000000119
4	18.7999992371	0.8000000119
5	25	0.8000000119
6	26.7000007629	0.8000000119
7	18.7999992371	0.8000000119
8	20	0.8000000119
9	18.7999992371	0.8000000119
10	25	0.8000000119
11	26.7000007629	0.8000000119
12	26.7000007629	0.8000000119

13	18.7999992371	0.8000000119
14	18.7999992371	0.8000000119
15	26.7000007629	0.8000000119
16	26.7000007629	0.8000000119
17	26.7000007629	0.8000000119
18	25	0.8000000119
19	37.5	0.8000000119
20	26.7000007629	0.8000000119
21	25	0.8000000119
22	20	0.8000000119
23	20	0.8000000119
24	25	0.8000000119
25	25	0.8000000119
26	20	0.8000000119
27	18.7999992371	0.8000000119
28	25	0.8000000119
29	26.7000007629	0.8000000119
30	20	0.8000000119
31	20	0.8000000119
32	18.7999992371	0.8000000119
33	20	0.8000000119
34	25	0.8000000119
35	25	0.8000000119
36	25	0.8000000119
37	25	0.8000000119
38	18.7999992371	0.8000000119
39	20	0.8000000119
40	25	0.8000000119
41	25	0.8000000119
42	18.7999992371	0.8000000119
43	18.7999992371	0.8000000119
44	18.7999992371	0.8000000119
45	26.7000007629	0.8000000119
46	33.2999992371	0.8000000119
47	25	0.8000000119
48	18.7999992371	0.8000000119
49	25	0.8000000119

Tabel E.3: Beads testresultaten van case 3.

Index	**%CPU**	**%Mem**
0	156.1999969482	0.8000000119
1	68.8000030518	0.8000000119
2	81.1999969482	0.8000000119
3	68.8000030518	0.8000000119
4	100	0.8000000119

5	68.8000030518	0.8000000119
6	62.5	0.8000000119
7	68.8000030518	0.8000000119
8	68.8000030518	0.8000000119
9	68.8000030518	0.8000000119
10	68.8000030518	0.8000000119
11	68.8000030518	0.8000000119
12	62.5	0.8000000119
13	68.8000030518	0.8000000119
14	68.8000030518	0.8000000119
15	66.6999969482	0.8000000119
16	68.8000030518	0.8000000119
17	68.8000030518	0.8000000119
18	70.5999984741	0.8000000119
19	68.8000030518	0.8000000119
20	68.8000030518	0.8000000119
21	68.8000030518	0.8000000119
22	75	0.8000000119
23	62.5	0.8000000119
24	62.5	0.8000000119
25	62.5	0.8000000119
26	62.5	0.8000000119
27	64.6999969482	0.8000000119
28	68.8000030518	0.8000000119
29	66.6999969482	0.8000000119
30	73.3000030518	0.8000000119
31	62.5	0.8000000119
32	62.5	0.8000000119
33	68.8000030518	0.8000000119
34	68.8000030518	0.8000000119
35	68.8000030518	0.8000000119
36	81.1999969482	0.8000000119
37	62.5	0.8000000119
38	68.8000030518	0.8000000119
39	62.5	0.8000000119
40	68.8000030518	0.8000000119
41	62.5	0.8000000119
42	68.8000030518	0.8000000119
43	75	0.8000000119
44	68.8000030518	0.8000000119
45	68.8000030518	0.8000000119
46	68.8000030518	0.8000000119
47	73.3000030518	0.8000000119
48	68.8000030518	0.8000000119
49	75	0.8000000119

Tabel E.4: Beads testresultaten van case 4.

Index	**%CPU**	**%Mem**
0	188.1999969482	1.1000000238
1	175	1.1000000238
2	81.1999969482	1.1000000238
3	87.5	1.1000000238
4	81.1999969482	1.1000000238
5	86.6999969482	1.1000000238
6	87.5	1.1000000238
7	93.8000030518	1.1000000238
8	81.1999969482	1.1000000238
9	93.3000030518	1.1000000238
10	87.5	1.1000000238
11	88.1999969482	1.1000000238
12	87.5	1.1000000238
13	93.8000030518	1.1000000238
14	87.5	1.1000000238
15	93.8000030518	1.1000000238
16	93.3000030518	1.1000000238
17	87.5	1.1000000238
18	93.3000030518	1.1000000238
19	87.5	1.1000000238
20	87.5	1.1000000238
21	87.5	1.1000000238
22	87.5	1.1000000238
23	87.5	1.1000000238
24	87.5	1.1000000238
25	93.8000030518	1.1000000238
26	87.5	1.1000000238
27	82.4000015259	1.1000000238
28	87.5	1.1000000238
29	87.5	1.1000000238
30	87.5	1.1000000238
31	87.5	1.1000000238
32	87.5	1.1000000238
33	86.6999969482	1.1000000238
34	81.1999969482	1.1000000238
35	87.5	1.1000000238
36	82.4000015259	1.1000000238
37	87.5	1.1000000238
38	87.5	1.1000000238
39	81.1999969482	1.1000000238
40	87.5	1.1000000238
41	81.1999969482	1.1000000238
42	56.2000007629	1.1000000238

43	87.5	1.1000000238
44	93.8000030518	1.1000000238
45	93.3000030518	1.1000000238
46	86.6999969482	1.1000000238
47	81.1999969482	1.1000000238
48	81.1999969482	1.1000000238
49	93.8000030518	1.1000000238

Tabel E.5: Beads testresultaten van case 5.

Index	**%CPU**	**%Mem**
0	87.5	1.5
1	87.5	1.5
2	87.5	1.5
3	87.5	1.5
4	93.8000030518	1.5
5	100	1.5
6	87.5	1.5
7	93.8000030518	1.5
8	93.8000030518	1.5
9	82.4000015259	1.5
10	81.1999969482	1.5
11	87.5	1.5
12	87.5	1.5
13	93.8000030518	1.5
14	100	1.5
15	87.5	1.5
16	93.8000030518	1.5
17	93.8000030518	1.5
18	93.8000030518	1.5
19	93.8000030518	1.5
20	93.8000030518	1.5
21	87.5	1.5
22	93.8000030518	1.5
23	87.5	1.5
24	93.8000030518	1.5
25	87.5	1.5
26	100	1.5
27	100	1.5
28	93.3000030518	1.5
29	100	1.5
30	100	1.5
31	87.5	1.5
32	93.8000030518	1.5
33	100	1.5
34	87.5	1.5

35	87.5	1.5
36	87.5	1.5
37	93.8000030518	1.5
38	87.5	1.5
39	87.5	1.5
40	87.5	1.5
41	87.5	1.5
42	100	1.5
43	87.5	1.5
44	93.8000030518	1.5
45	87.5	1.5
46	87.5	1.5
47	87.5	1.5
48	93.8000030518	1.5
49	93.8000030518	1.5

Tabel E.6: Beads testresultaten van case 6.

Index	**%CPU**	**%Mem**
0	87.5	1.1000000238
1	81.1999969482	1.1000000238
2	86.6999969482	1.1000000238
3	93.3000030518	1.1000000238
4	93.3000030518	1.1000000238
5	93.8000030518	1.1000000238
6	87.5	1.1000000238
7	86.6999969482	1.1000000238
8	87.5	1.1000000238
9	82.4000015259	1.1000000238
10	87.5	1.1000000238
11	93.3000030518	1.1000000238
12	87.5	1.1000000238
13	93.8000030518	1.1000000238
14	81.1999969482	1.1000000238
15	87.5	1.1000000238
16	87.5	1.1000000238
17	87.5	1.1000000238
18	87.5	1.1000000238
19	87.5	1.1000000238
20	87.5	1.1000000238
21	87.5	1.1000000238
22	87.5	1.1000000238
23	87.5	1.1000000238
24	87.5	1.1000000238
25	87.5	1.1000000238
26	87.5	1.1000000238

27	93.8000030518	1.1000000238
28	87.5	1.1000000238
29	81.1999969482	1.1000000238
30	87.5	1.1000000238
31	87.5	1.1000000238
32	87.5	1.1000000238
33	87.5	1.1000000238
34	81.1999969482	1.1000000238
35	81.1999969482	1.1000000238
36	87.5	1.1000000238
37	81.1999969482	1.1000000238
38	87.5	1.1000000238
39	86.6999969482	1.1000000238
40	82.4000015259	1.1000000238
41	82.4000015259	1.1000000238
42	87.5	1.1000000238
43	81.1999969482	1.1000000238
44	81.1999969482	1.1000000238
45	87.5	1.1000000238
46	81.1999969482	1.1000000238
47	81.1999969482	1.1000000238
48	81.1999969482	1.1000000238
49	87.5	1.1000000238

E.2 Testresultaten van JASS

Tabel E.7: JASS testresultaten van case 1.

Index	**%CPU**	**%Mem**
0	125	0.8999999762
1	87.5	0.8999999762
2	93.8000030518	0.8999999762
3	93.8000030518	0.8999999762
4	100	0.8999999762
5	93.3000030518	0.8999999762
6	100	0.8999999762
7	93.8000030518	0.8999999762
8	93.8000030518	0.8999999762
9	93.8000030518	0.8999999762
10	93.8000030518	0.8999999762
11	93.8000030518	0.8999999762
12	88.1999969482	0.8999999762
13	100	0.8999999762
14	93.8000030518	0.8999999762
15	100	0.8999999762
16	93.8000030518	0.8999999762

17	93.3000030518	0.8999999762
18	93.3000030518	0.8999999762
19	87.5	0.8999999762
20	87.5	0.8999999762
21	93.8000030518	0.8999999762
22	61.9000015259	0.8999999762
23	73.6999969482	0.8999999762
24	47.0999984741	0.8999999762
25	43.7999992371	0.8999999762
26	87.5	0.8999999762
27	93.8000030518	0.8999999762
28	112.5	0.8999999762
29	193.8000030518	0.8999999762
30	93.8000030518	0.8999999762
31	100	0.8999999762
32	93.8000030518	0.8999999762
33	88.1999969482	0.8999999762
34	88.1999969482	0.8999999762
35	87.5	0.8999999762
36	143.8000030518	0.8999999762
37	181.1999969482	1
38	162.5	1
39	187.5	1
40	100	1
41	87.5	1
42	93.8000030518	1
43	82.4000015259	1
44	93.8000030518	1
45	87.5	1
46	100	1
47	93.8000030518	1
48	93.8000030518	1
49	87.5	1

Tabel E.8: JASS testresultaten van case 2.

Index	**%CPU**	**%Mem**
0	131.1999969482	1
1	168.8000030518	1
2	193.8000030518	1
3	100	1
4	100	1
5	93.8000030518	1
6	93.8000030518	1
7	93.8000030518	1
8	93.8000030518	1

9	93.8000030518	1
10	93.8000030518	1
11	93.8000030518	1
12	100	1
13	93.8000030518	1
14	93.8000030518	1
15	93.8000030518	1
16	100	1
17	93.8000030518	1
18	93.8000030518	1
19	100	1
20	100	1
21	93.8000030518	1
22	93.8000030518	1
23	87.5	1
24	100	1
25	93.8000030518	1
26	100	1
27	93.8000030518	1
28	93.8000030518	1
29	93.8000030518	1
30	100	1
31	93.8000030518	1
32	93.8000030518	1
33	93.8000030518	1
34	93.8000030518	1
35	93.8000030518	1
36	100	1
37	93.8000030518	1
38	87.5	1
39	100	1
40	93.8000030518	1
41	93.8000030518	1
42	100	1
43	93.8000030518	1
44	106.6999969482	1
45	93.8000030518	1
46	93.8000030518	1
47	100	1
48	93.8000030518	1
49	93.8000030518	1

Tabel E.9: JASS testresultaten van case 3.

Index	**%CPU**	**%Mem**
0	100	1

1	93.8000030518	1
2	93.8000030518	1
3	93.8000030518	1.1000000238
4	93.8000030518	1.1000000238
5	93.8000030518	1.1000000238
6	93.8000030518	1.1000000238
7	106.1999969482	1.1000000238
8	93.8000030518	1.1000000238
9	100	1.1000000238
10	100	1.1000000238
11	100	1.1000000238
12	100	1.1000000238
13	100	1.1000000238
14	106.6999969482	1.1000000238
15	100	1.1000000238
16	100	1.1000000238
17	106.6999969482	1.1000000238
18	113.3000030518	1.1000000238
19	112.5	1.1000000238
20	93.8000030518	1.1000000238
21	93.8000030518	1.1000000238
22	93.8000030518	1.1000000238
23	100	1.1000000238
24	93.8000030518	1.1000000238
25	100	1.1000000238
26	100	1.1000000238
27	100	1.1000000238
28	100	1.1000000238
29	94.0999984741	1.1000000238
30	100	1.1000000238
31	100	1.1000000238
32	106.6999969482	1.1000000238
33	100	1.1000000238
34	93.3000030518	1.1000000238
35	100	1.1000000238
36	100	1.1000000238
37	106.6999969482	1.1000000238
38	100	1.1000000238
39	93.8000030518	1.1000000238
40	106.6999969482	1.1000000238
41	93.8000030518	1.1000000238
42	106.6999969482	1.1000000238
43	93.8000030518	1.1000000238
44	100	1.1000000238
45	100	1.1000000238
46	93.8000030518	1.1000000238

47	100	1.1000000238
48	106.6999969482	1.1000000238
49	100	1.1000000238

Tabel E.10: JASS testresultaten van case 4.

Index	**%CPU**	**%Mem**
0	106.1999969482	1.2000000477
1	100	1.2000000477
2	100	1.2000000477
3	100	1.2000000477
4	100	1.2000000477
5	106.6999969482	1.2000000477
6	93.8000030518	1.2000000477
7	131.1999969482	1.2000000477
8	100	1.2000000477
9	93.8000030518	1.2000000477
10	100	1.2000000477
11	100	1.2000000477
12	106.6999969482	1.2000000477
13	100	1.2000000477
14	100	1.2000000477
15	100	1.2000000477
16	93.8000030518	1.2000000477
17	100	1.2000000477
18	100	1.2000000477
19	93.8000030518	1.2000000477
20	100	1.2000000477
21	100	1.2000000477
22	93.8000030518	1.2000000477
23	100	1.2000000477
24	106.6999969482	1.2000000477
25	106.6999969482	1.2000000477
26	93.8000030518	1.2000000477
27	100	1.2000000477
28	100	1.2000000477
29	93.8000030518	1.2000000477
30	100	1.2000000477
31	100	1.2000000477
32	100	1.2000000477
33	106.6999969482	1.2000000477
34	93.8000030518	1.2000000477
35	93.8000030518	1.2000000477
36	93.8000030518	1.2000000477
37	100	1.2000000477
38	106.1999969482	1.2000000477

39	106.6999969482	1.2000000477
40	100	1.2000000477
41	106.6999969482	1.2000000477
42	100	1.2000000477
43	106.6999969482	1.2000000477
44	93.8000030518	1.2000000477
45	100	1.2000000477
46	100	1.2000000477
47	93.8000030518	1.2000000477
48	100	1.2000000477
49	100	1.2000000477

Tabel E.11: JASS testresultaten van case 5.

Index	**%CPU**	**%Mem**
0	100	1.2999999523
1	93.8000030518	1.2999999523
2	100	1.2999999523
3	100	1.2999999523
4	100	1.2999999523
5	93.8000030518	1.2999999523
6	100	1.2999999523
7	93.8000030518	1.2999999523
8	100	1.2999999523
9	100	1.2999999523
10	100	1.2999999523
11	93.8000030518	1.2999999523
12	93.8000030518	1.2999999523
13	112.5	1.2999999523
14	100	1.2999999523
15	106.6999969482	1.2999999523
16	93.8000030518	1.2999999523
17	93.8000030518	1.2999999523
18	93.8000030518	1.2999999523
19	100	1.2999999523
20	106.6999969482	1.2999999523
21	100	1.2999999523
22	93.8000030518	1.2999999523
23	106.6999969482	1.2999999523
24	106.6999969482	1.2999999523
25	100	1.2999999523
26	93.8000030518	1.2999999523
27	100	1.2999999523
28	100	1.2999999523
29	100	1.2999999523
30	93.8000030518	1.2999999523

31	100	1.2999999523
32	100	1.2999999523
33	100	1.2999999523
34	93.8000030518	1.2999999523
35	100	1.2999999523
36	106.6999969482	1.2999999523
37	100	1.2999999523
38	100	1.2999999523
39	93.8000030518	1.2999999523
40	93.8000030518	1.2999999523
41	93.8000030518	1.2999999523
42	100	1.2999999523
43	100	1.2999999523
44	100	1.2999999523
45	100	1.2999999523
46	100	1.2999999523
47	100	1.2999999523
48	100	1.3999999762
49	93.8000030518	1.3999999762

Tabel E.12: JASS testresultaten van case 6.

Index	**%CPU**	**%Mem**
0	93.8000030518	1.8999999762
1	93.8000030518	1.8999999762
2	100	1.8999999762
3	100	1.8999999762
4	100	1.8999999762
5	93.8000030518	1.8999999762
6	93.8000030518	1.8999999762
7	100	1.8999999762
8	93.8000030518	1.8999999762
9	100	1.8999999762
10	100	1.8999999762
11	100	1.8999999762
12	106.6999969482	1.8999999762
13	100	1.8999999762
14	100	1.8999999762
15	100	1.8999999762
16	100	1.8999999762
17	100	1.8999999762
18	106.6999969482	1.8999999762
19	100	1.8999999762
20	100	1.8999999762
21	106.1999969482	1.8999999762
22	93.8000030518	1.8999999762

23	100	1.8999999762
24	100	1.8999999762
25	93.8000030518	1.8999999762
26	93.8000030518	1.8999999762
27	100	1.8999999762
28	100	1.8999999762
29	106.6999969482	1.8999999762
30	94.0999984741	1.8999999762
31	100	1.8999999762
32	106.6999969482	1.8999999762
33	106.6999969482	1.8999999762
34	93.8000030518	1.8999999762
35	100	1.8999999762
36	100	1.8999999762
37	93.8000030518	1.8999999762
38	100	1.8999999762
39	100	1.8999999762
40	100	1.8999999762
41	100	1.8999999762
42	100	1.8999999762
43	100	1.8999999762
44	100	1.8999999762
45	106.6999969482	1.8999999762
46	100	1.8999999762
47	93.8000030518	1.8999999762
48	100	1.8999999762
49	100	1.8999999762

E.3 Testresultaten van JSyn

Tabel E.13: JSyn testresultaten van case 1.

Index	**%CPU**	**%Mem**
0	137.5	2.0999999046
1	212.5	2.0999999046
2	206.1999969482	2.0999999046
3	131.1999969482	2.0999999046
4	25	2.0999999046
5	18.7999992371	2.0999999046
6	13.3000001907	2.0999999046
7	25	2.0999999046
8	6.1999998093	2.0999999046
9	12.5	2.0999999046
10	6.1999998093	2.0999999046
11	13.3000001907	2.0999999046

12	12.5	2.0999999046
13	6.1999998093	2.0999999046
14	6.6999998093	2.0999999046
15	12.5	2.0999999046
16	6.1999998093	2.0999999046
17	40	2.0999999046
18	6.1999998093	2.0999999046
19	6.1999998093	2.0999999046
20	12.5	2.0999999046
21	12.5	2.0999999046
22	12.5	2.0999999046
23	13.3000001907	2.0999999046
24	12.5	2.0999999046
25	6.6999998093	2.0999999046
26	6.1999998093	2.0999999046
27	6.6999998093	2.0999999046
28	20	2.0999999046
29	12.5	2.0999999046
30	6.1999998093	2.0999999046
31	25	2.0999999046
32	13.3000001907	2.0999999046
33	6.6999998093	2.0999999046
34	6.1999998093	2.0999999046
35	18.7999992371	2.0999999046
36	6.6999998093	2.0999999046
37	12.5	2.0999999046
38	13.3000001907	2.0999999046
39	6.1999998093	2.0999999046
40	6.1999998093	2.0999999046
41	6.1999998093	2.0999999046
42	13.3000001907	2.0999999046
43	6.1999998093	2.0999999046
44	20	2.0999999046
45	12.5	2.0999999046
46	13.3000001907	2.0999999046
47	12.5	2.0999999046
48	12.5	2.0999999046
49	12.5	2.0999999046

Tabel E.14: JSyn testresultaten van case 2.

Index	**%CPU**	**%Mem**
0	180	2.2000000477
1	218.8000030518	2.2000000477
2	218.8000030518	2.2999999523
3	81.1999969482	2.2999999523

4	18.7999992371	2.2999999523
5	18.7999992371	2.2999999523
6	18.7999992371	2.2999999523
7	12.5	2.2999999523
8	31.2000007629	2.2999999523
9	18.7999992371	2.2999999523
10	25	2.2999999523
11	20	2.2999999523
12	12.5	2.2999999523
13	25	2.2999999523
14	18.7999992371	2.2999999523
15	18.7999992371	2.2999999523
16	20	2.2999999523
17	25	2.2999999523
18	12.5	2.2999999523
19	62.5	2.2999999523
20	25	2.2999999523
21	18.7999992371	2.2999999523
22	18.7999992371	2.2999999523
23	20	2.2999999523
24	25	2.2999999523
25	12.5	2.2999999523
26	18.7999992371	2.2999999523
27	20	2.2999999523
28	18.7999992371	2.2999999523
29	20	2.2999999523
30	18.7999992371	2.2999999523
31	12.5	2.2999999523
32	13.3000001907	2.2999999523
33	18.7999992371	2.2999999523
34	13.3000001907	2.2999999523
35	12.5	2.2999999523
36	20	2.2999999523
37	18.7999992371	2.2999999523
38	20	2.2999999523
39	18.7999992371	2.2999999523
40	20	2.2999999523
41	18.7999992371	2.2999999523
42	25	2.2999999523
43	18.7999992371	2.2999999523
44	18.7999992371	2.2999999523
45	12.5	2.2999999523
46	12.5	2.2999999523
47	18.7999992371	2.2999999523
48	18.7999992371	2.2999999523
49	25	2.2999999523

Tabel E.15: JSyn testresultaten van case 3.

Index	**%CPU**	**%Mem**
0	37.5	2.2999999523
1	31.2000007629	2.2999999523
2	31.2000007629	2.2999999523
3	33.2999992371	2.2999999523
4	37.5	2.2999999523
5	31.2000007629	2.2999999523
6	46.7000007629	2.2999999523
7	31.2000007629	2.2999999523
8	37.5	2.2999999523
9	33.2999992371	2.2999999523
10	26.7000007629	2.2999999523
11	43.7999992371	2.2999999523
12	37.5	2.2999999523
13	33.2999992371	2.2999999523
14	37.5	2.2999999523
15	31.2000007629	2.2999999523
16	40	2.2999999523
17	33.2999992371	2.2999999523
18	33.2999992371	2.2999999523
19	43.7999992371	2.2999999523
20	31.2000007629	2.2999999523
21	43.7999992371	2.2999999523
22	33.2999992371	2.2999999523
23	31.2000007629	2.2999999523
24	33.2999992371	2.2999999523
25	31.2000007629	2.2999999523
26	43.7999992371	2.2999999523
27	31.2000007629	2.2999999523
28	31.2000007629	2.2999999523
29	37.5	2.2999999523
30	37.5	2.2999999523
31	37.5	2.2999999523
32	46.7000007629	2.2999999523
33	37.5	2.2999999523
34	33.2999992371	2.2999999523
35	31.2000007629	2.2999999523
36	31.2000007629	2.2999999523
37	31.2000007629	2.2999999523
38	37.5	2.2999999523
39	31.2000007629	2.2999999523
40	33.2999992371	2.2999999523
41	31.2000007629	2.2999999523
42	31.2000007629	2.2999999523

43	31.2000007629	2.2999999523
44	25	2.2999999523
45	40	2.2999999523
46	37.5	2.2999999523
47	37.5	2.2999999523
48	40	2.2999999523
49	37.5	2.2999999523

Tabel E.16: JSyn testresultaten van case 4.

Index	**%CPU**	**%Mem**
0	50	2.2999999523
1	56.2000007629	2.2999999523
2	56.2000007629	2.2999999523
3	53.2999992371	2.2999999523
4	56.2000007629	2.2999999523
5	50	2.2999999523
6	56.2000007629	2.2999999523
7	56.2000007629	2.2999999523
8	53.2999992371	2.2999999523
9	56.2000007629	2.2999999523
10	50	2.2999999523
11	50	2.2999999523
12	53.2999992371	2.2999999523
13	60	2.2999999523
14	50	2.2999999523
15	56.2000007629	2.2999999523
16	56.2000007629	2.2999999523
17	66.6999969482	2.2999999523
18	53.2999992371	2.2999999523
19	50	2.2999999523
20	56.2000007629	2.2999999523
21	56.2000007629	2.2999999523
22	68.8000030518	2.2999999523
23	50	2.2999999523
24	60	2.2999999523
25	53.2999992371	2.2999999523
26	29.3999996185	2.2999999523
27	27.7999992371	2.2999999523
28	23.5	2.2999999523
29	27.7999992371	2.2999999523
30	22.2000007629	2.2999999523
31	26.2999992371	2.2999999523
32	37.5	2.2999999523
33	36.7999992371	2.2999999523
34	29.3999996185	2.2999999523

35	41.2000007629	2.2999999523
36	31.2000007629	2.2999999523
37	31.2000007629	2.2999999523
38	31.2000007629	2.2999999523
39	31.2000007629	2.2999999523
40	29.3999996185	2.2999999523
41	31.2000007629	2.2999999523
42	33.2999992371	2.2999999523
43	21.1000003815	2.2999999523
44	37.5	2.2999999523
45	23.5	2.2999999523
46	35.2999992371	2.2999999523
47	37.5	2.2999999523
48	37.5	2.2999999523
49	37.5	2.2999999523

Tabel E.17: JSyn testresultaten van case 5.

Index	**%CPU**	**%Mem**
0	56.2000007629	2.2999999523
1	50	2.2999999523
2	47.0999984741	2.2999999523
3	41.2000007629	2.2999999523
4	43.7999992371	2.2999999523
5	37.5	2.2999999523
6	37.5	2.2999999523
7	50	2.2999999523
8	50	2.2999999523
9	43.7999992371	2.2999999523
10	43.7999992371	2.2999999523
11	43.7999992371	2.2999999523
12	50	2.2999999523
13	62.5	2.2999999523
14	75	2.2999999523
15	68.8000030518	2.2999999523
16	70.5999984741	2.2999999523
17	68.8000030518	2.2999999523
18	68.8000030518	2.2999999523
19	68.8000030518	2.2999999523
20	68.8000030518	2.2999999523
21	68.8000030518	2.2999999523
22	68.8000030518	2.2999999523
23	94.0999984741	2.2999999523
24	75	2.2999999523
25	68.8000030518	2.2999999523
26	68.8000030518	2.2999999523

27	75	2.2999999523
28	75	2.2999999523
29	75	2.2999999523
30	62.5	2.2999999523
31	75	2.2999999523
32	80	2.2999999523
33	68.8000030518	2.2999999523
34	75	2.2999999523
35	68.8000030518	2.2999999523
36	68.8000030518	2.2999999523
37	68.8000030518	2.2999999523
38	81.1999969482	2.2999999523
39	68.8000030518	2.2999999523
40	68.8000030518	2.2999999523
41	73.3000030518	2.2999999523
42	68.8000030518	2.2999999523
43	73.3000030518	2.2999999523
44	75	2.2999999523
45	68.8000030518	2.2999999523
46	62.5	2.2999999523
47	75	2.2999999523
48	62.5	2.2999999523
49	75	2.2999999523

Tabel E.18: JSyn testresultaten van case 6.

Index	**%CPU**	**%Mem**
0	75	2.2999999523
1	56.2000007629	2.2999999523
2	62.5	2.2999999523
3	56.2000007629	2.2999999523
4	50	2.2999999523
5	62.5	2.2999999523
6	56.2000007629	2.2999999523
7	58.7999992371	2.2999999523
8	50	2.2999999523
9	50	2.2999999523
10	53.2999992371	2.2999999523
11	62.5	2.2999999523
12	56.2000007629	2.2999999523
13	56.2000007629	2.2999999523
14	56.2000007629	2.2999999523
15	62.5	2.2999999523
16	58.7999992371	2.2999999523
17	56.2000007629	2.2999999523
18	56.2000007629	2.2999999523

19	56.2000007629	2.2999999523
20	56.2000007629	2.2999999523
21	62.5	2.2999999523
22	56.2000007629	2.2999999523
23	50	2.2999999523
24	56.2000007629	2.2999999523
25	50	2.2999999523
26	56.2000007629	2.2999999523
27	62.5	2.2999999523
28	43.7999992371	2.2999999523
29	56.2000007629	2.2999999523
30	56.2000007629	2.2999999523
31	60	2.2999999523
32	56.2000007629	2.2999999523
33	56.2000007629	2.2999999523
34	60	2.2999999523
35	56.2000007629	2.2999999523
36	52.9000015259	2.2999999523
37	56.2000007629	2.2999999523
38	56.2000007629	2.2999999523
39	62.5	2.2999999523
40	50	2.2999999523
41	56.2000007629	2.2999999523
42	56.2000007629	2.2999999523
43	50	2.2999999523
44	50	2.2999999523
45	56.2000007629	2.2999999523
46	50	2.2999999523
47	52.9000015259	2.2999999523
48	56.2000007629	2.2999999523
49	62.5	2.2999999523

F. Smoothing van de Testresultaten

F.1 Smoothing van CPU testresultaten voor Beads

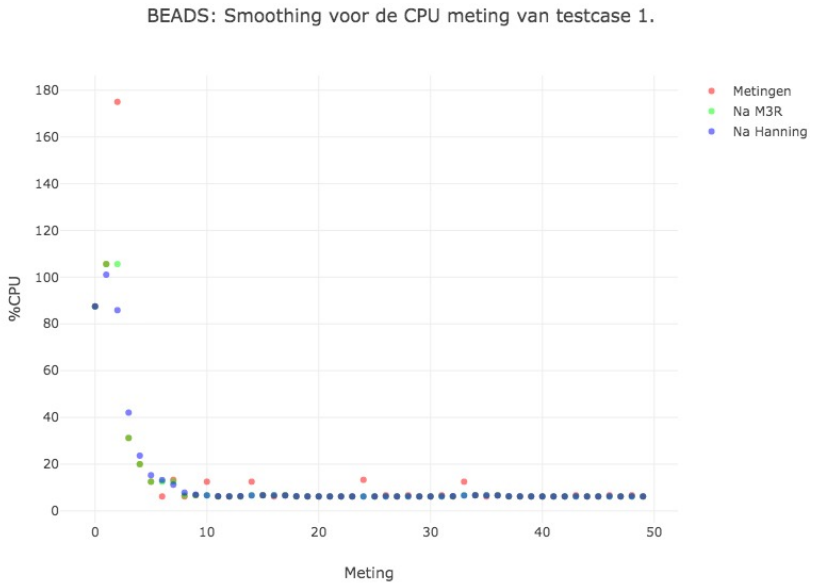
F.2 Smoothing van Mem testresultaten voor Beads

F.3 Smoothing van CPU testresultaten voor JASS

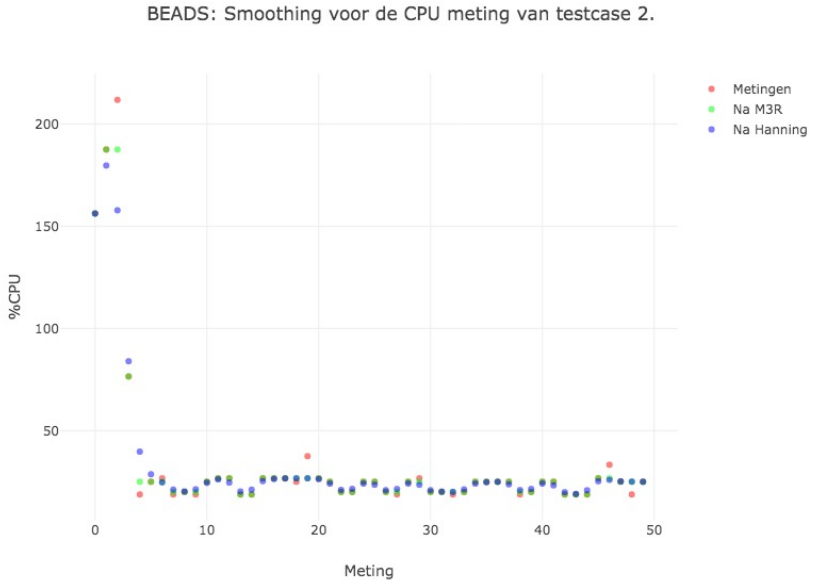
F.4 Smoothing van Mem testresultaten voor JASS

F.5 Smoothing van CPU testresultaten voor JSyn

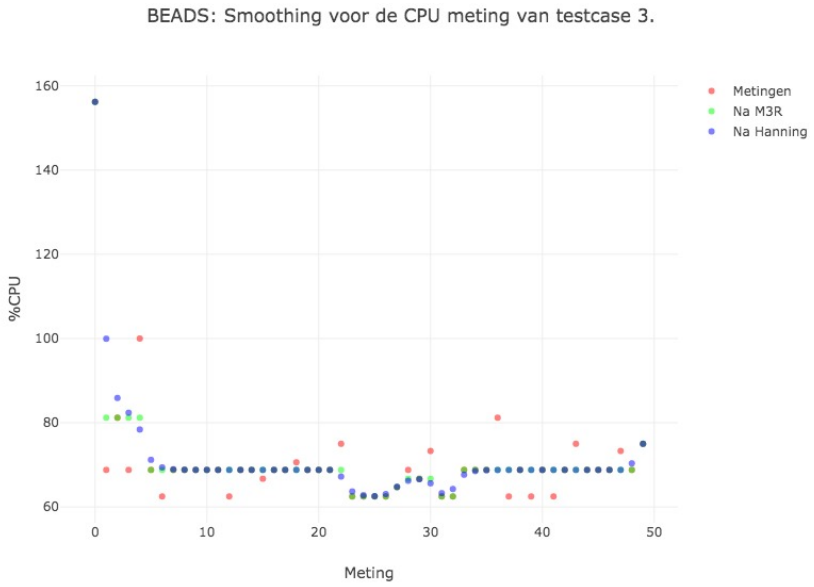
F.6 Smoothing van Mem testresultaten voor JSyn



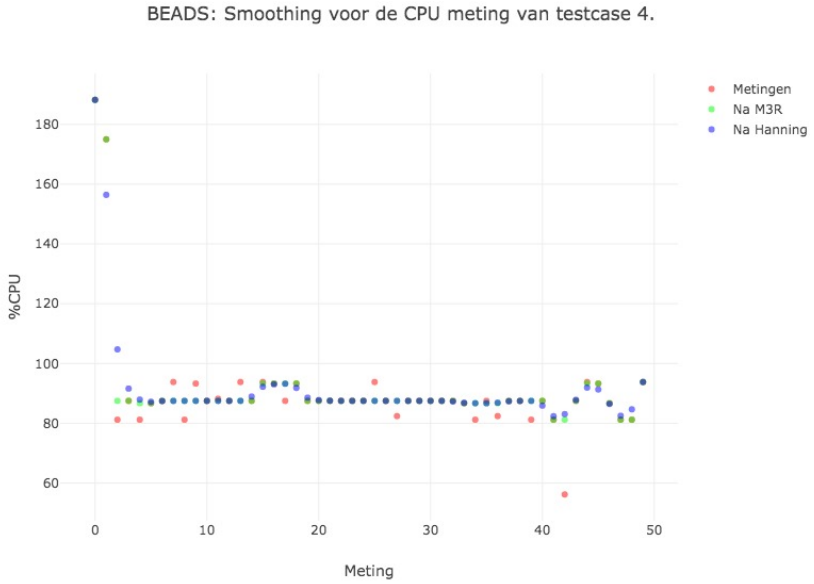
Figuur F.1: Smoothing van CPU testresultaten van testcase 1voor Beads



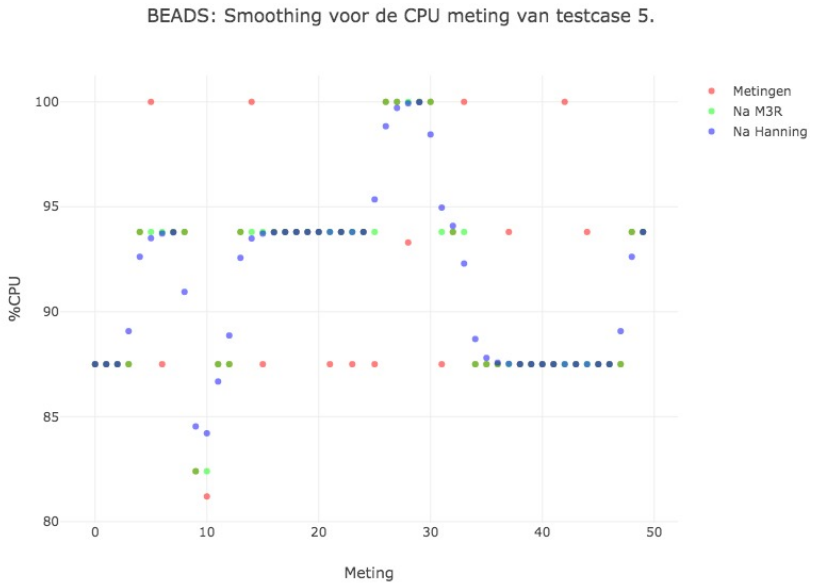
Figuur F.2: Smoothing van CPU testresultaten van testcase 2voor Beads



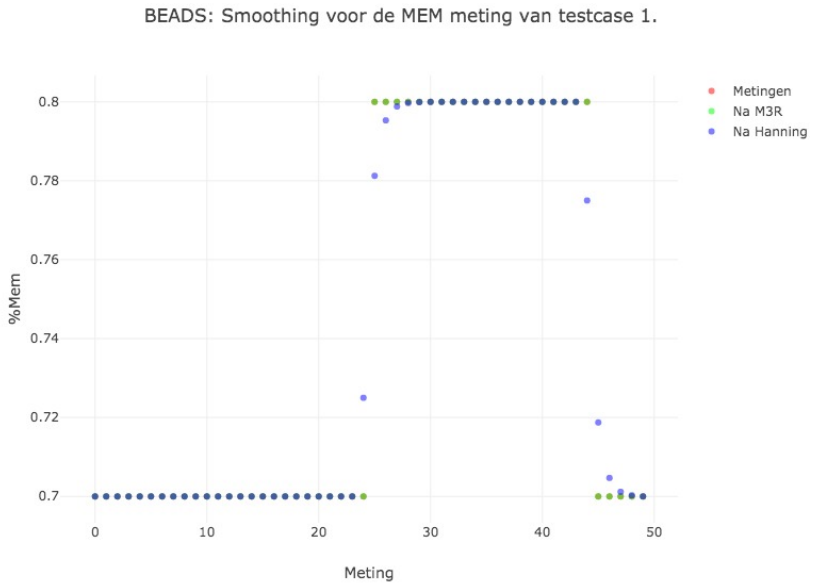
Figuur F.3: Smoothing van CPU testresultaten van testcase 3voor Beads



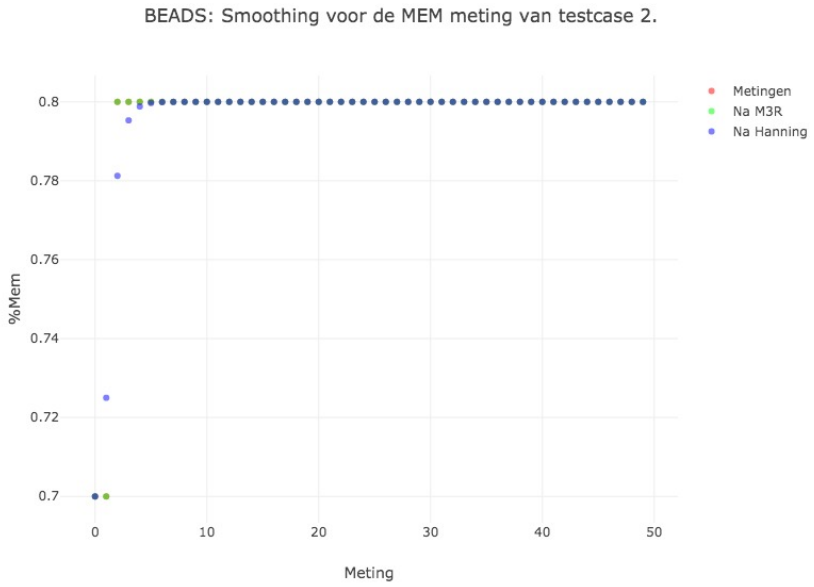
Figuur F.4: Smoothing van CPU testresultaten van testcase 4voor Beads



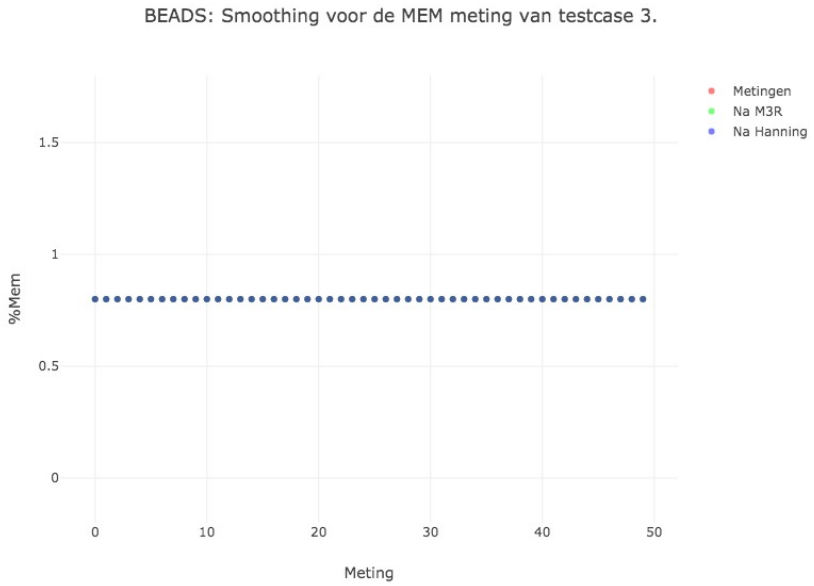
Figuur F.5: Smoothing van CPU testresultaten van testcase 5voor Beads



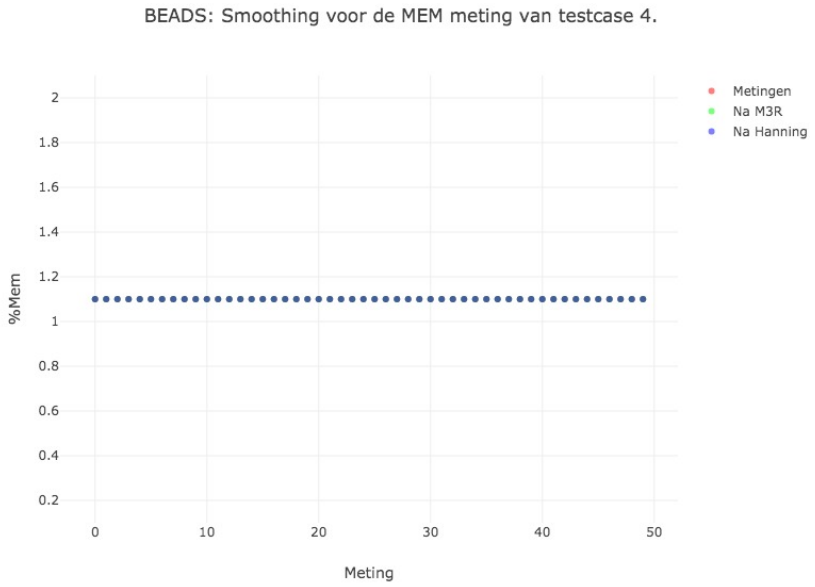
Figuur F.6: Smoothing van Mem testresultaten van testcase 1voor Beads



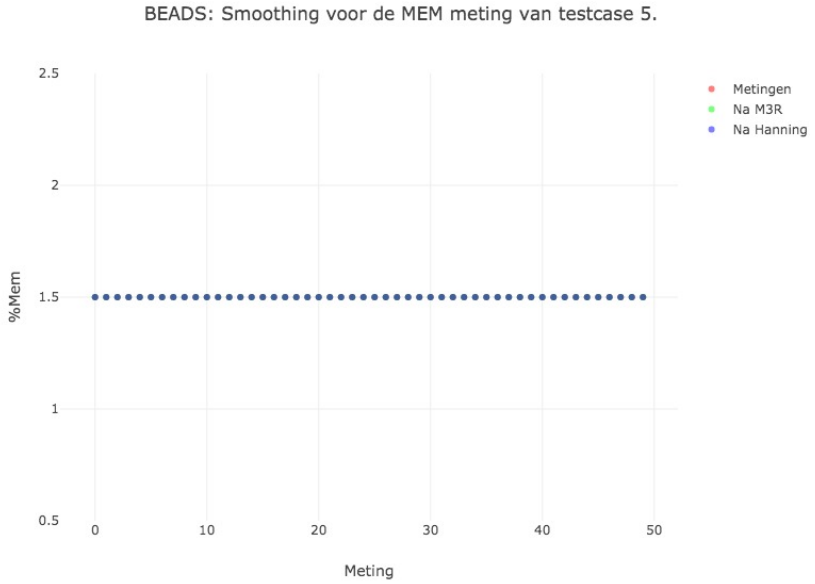
Figuur F.7: Smoothing van Mem testresultaten van testcase 2voor Beads



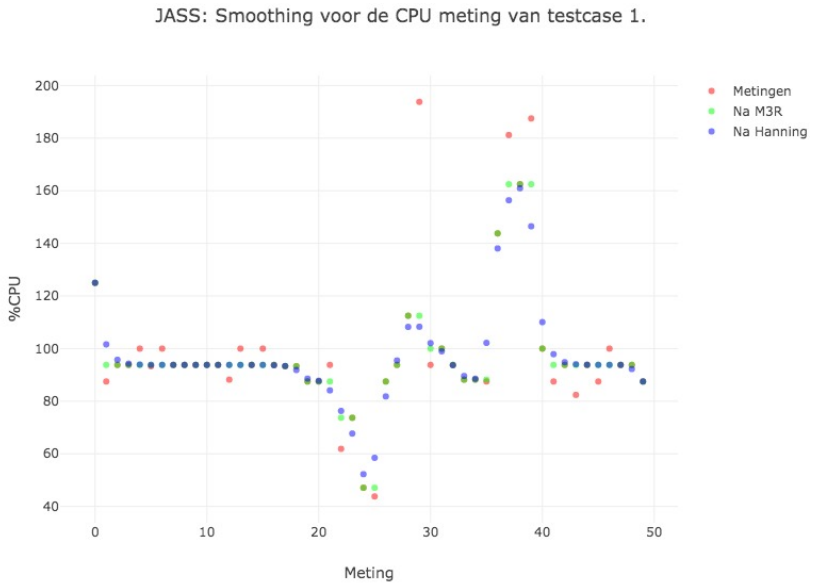
Figuur F.8: Smoothing van Mem testresultaten van testcase 3voor Beads



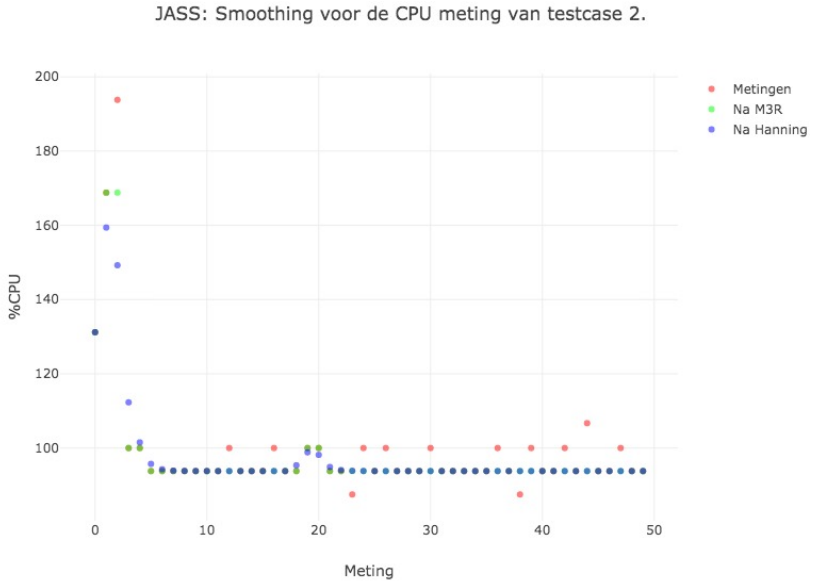
Figuur F.9: Smoothing van Mem testresultaten van testcase 4voor Beads



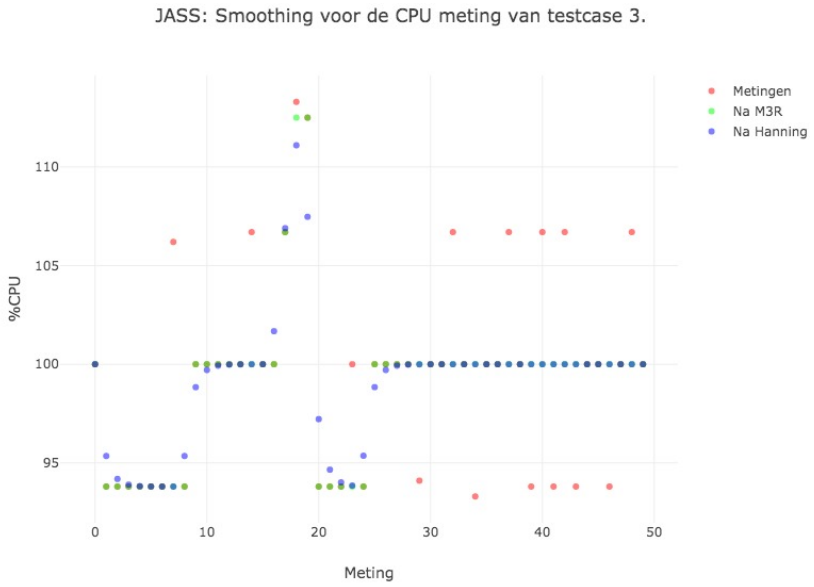
Figuur F.10: Smoothing van Mem testresultaten van testcase 5voor Beads



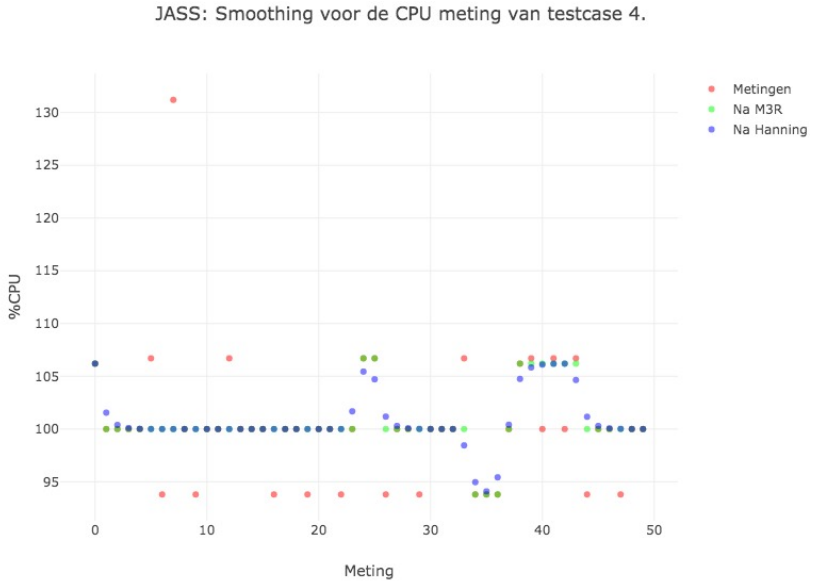
Figuur F.11: Smoothing van CPU testresultaten van testcase 1voor JASS



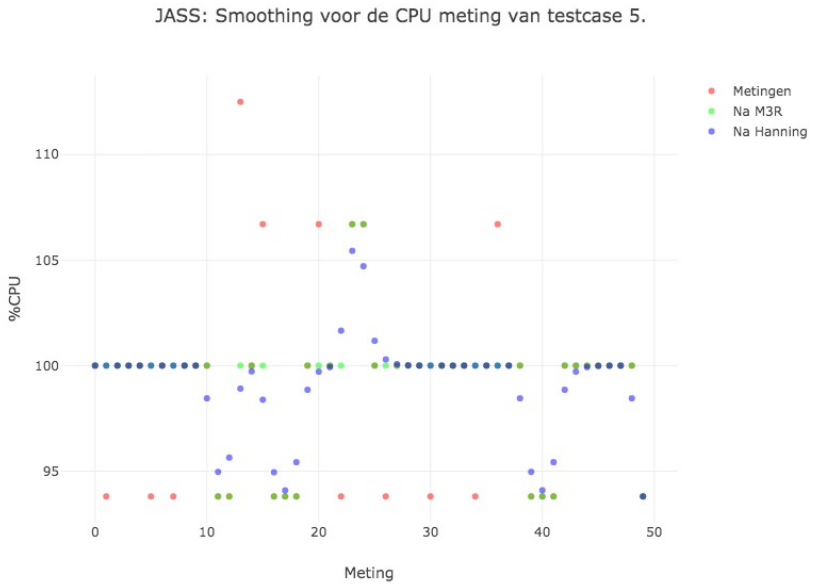
Figuur F.12: Smoothing van CPU testresultaten van testcase 2voor JASS



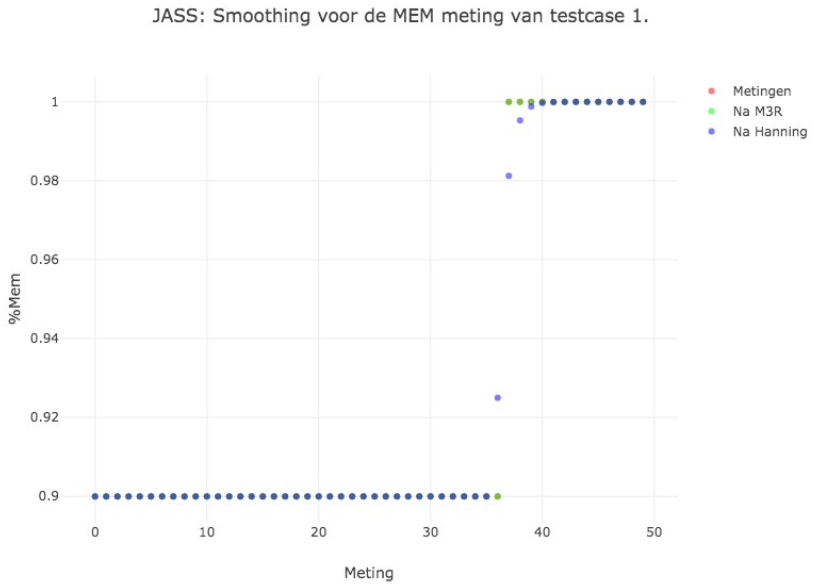
Figuur F.13: Smoothing van CPU testresultaten van testcase 3voor JASS



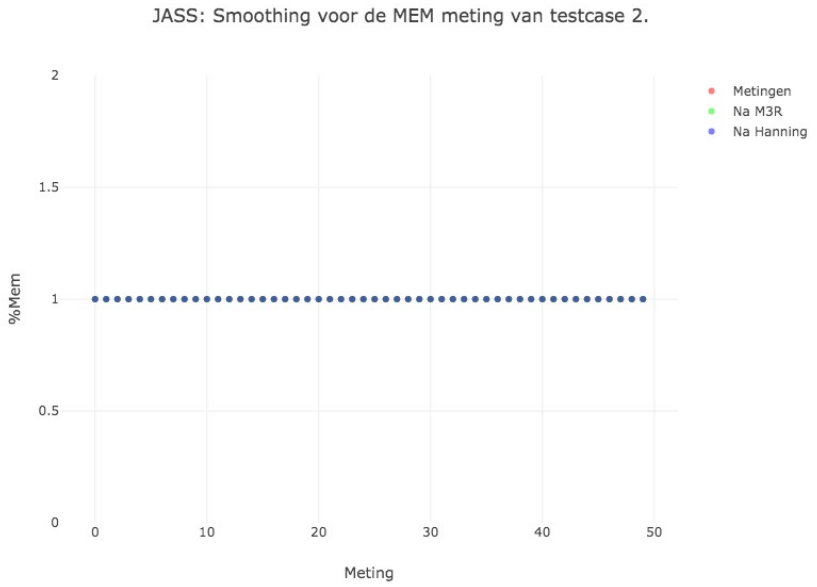
Figuur F.14: Smoothing van CPU testresultaten van testcase 4voor JASS



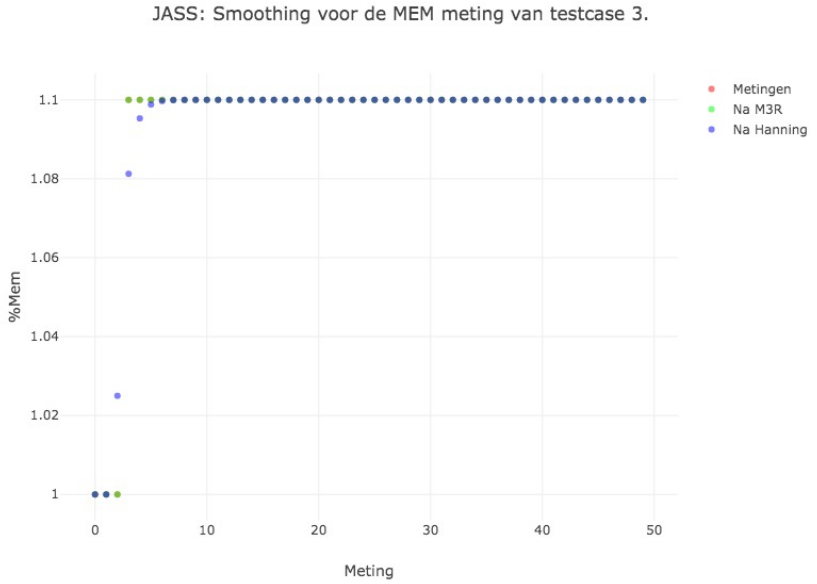
Figuur F.15: Smoothing van CPU testresultaten van testcase 5voor JASS



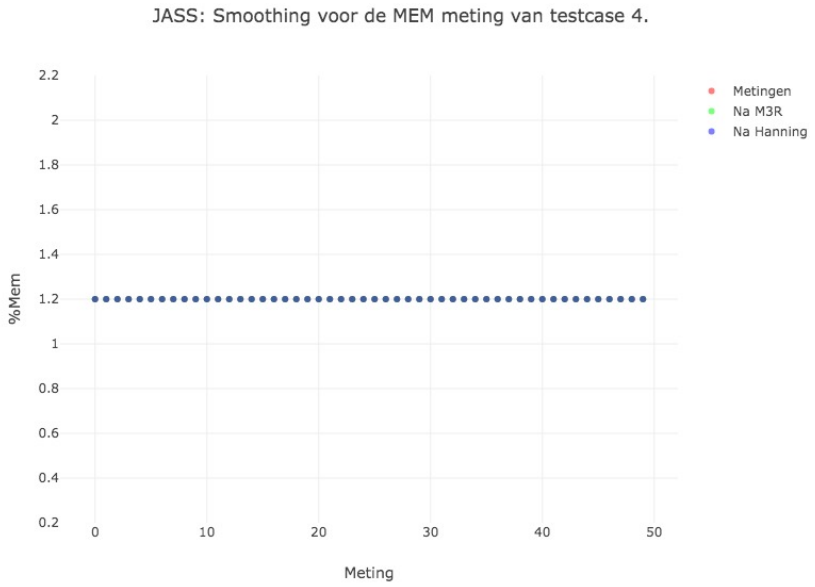
Figuur F.16: Smoothing van Mem testresultaten van testcase 1voor JASS



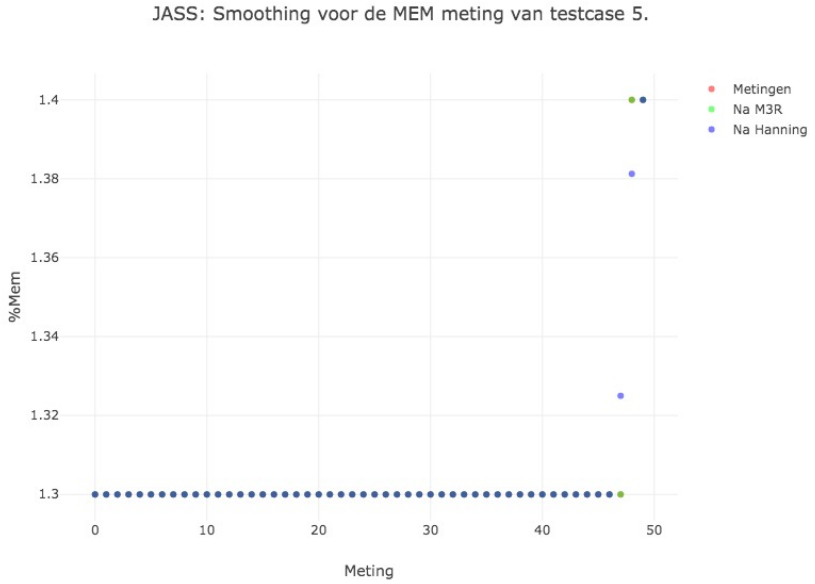
Figuur F.17: Smoothing van Mem testresultaten van testcase 2voor JASS



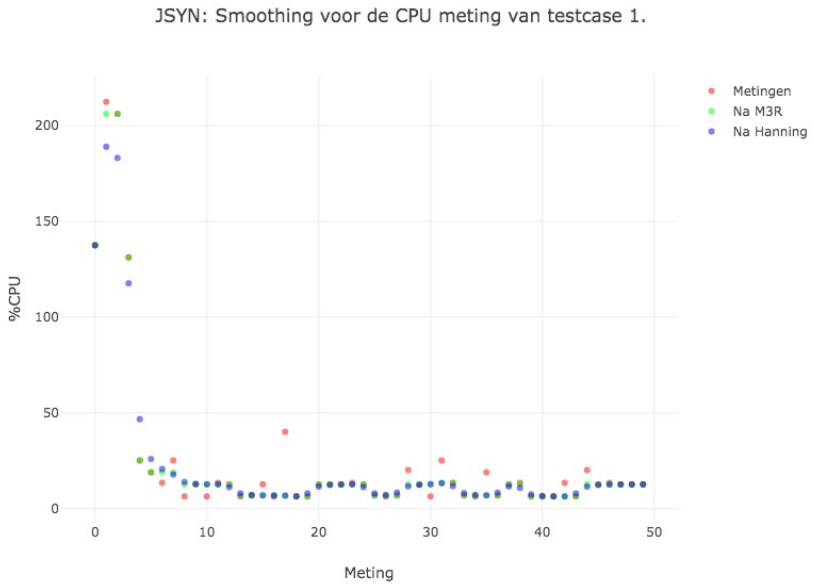
Figuur F.18: Smoothing van Mem testresultaten van testcase 3voor JASS



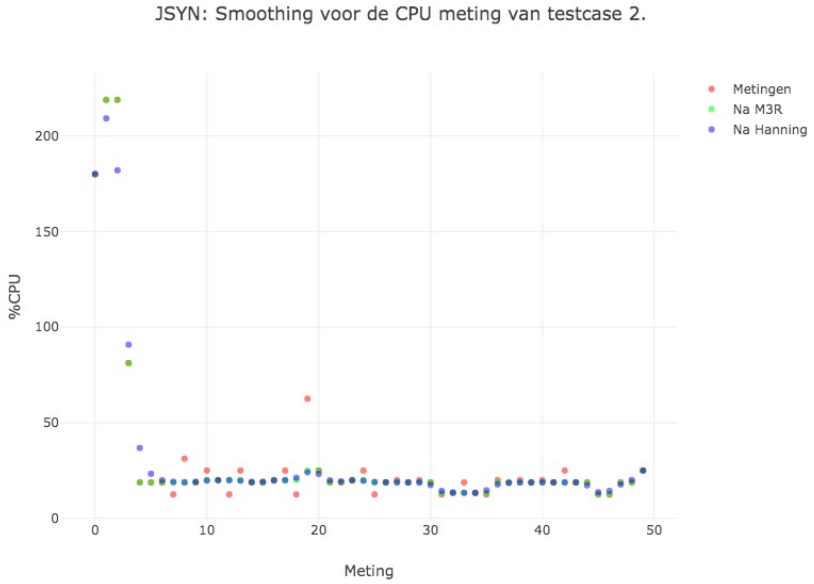
Figuur F.19: Smoothing van Mem testresultaten van testcase 4voor JASS



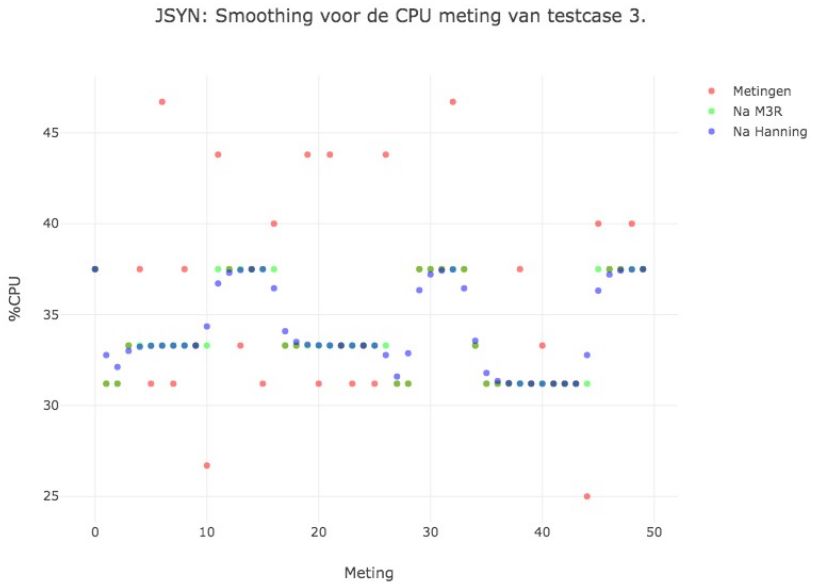
Figuur F.20: Smoothing van Mem testresultaten van testcase 5voor JASS



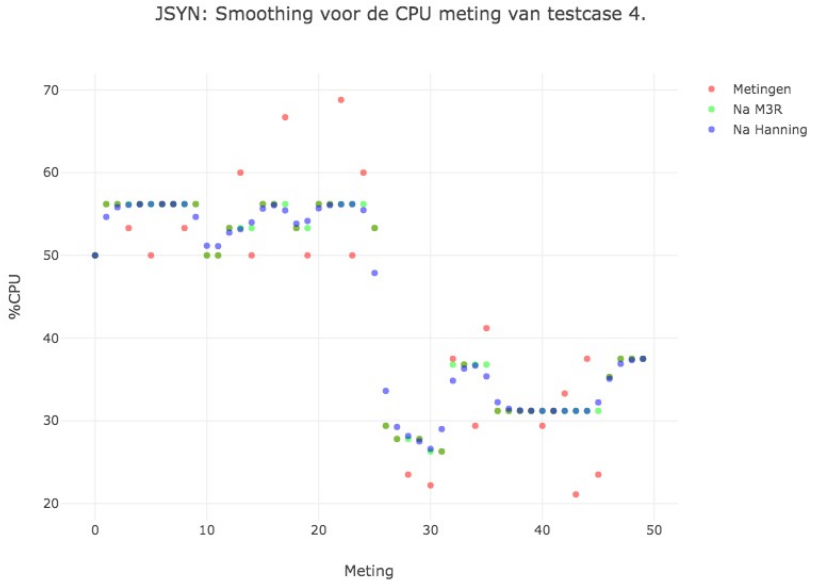
Figuur F.21: Smoothing van CPU testresultaten van testcase 1voor JSyn



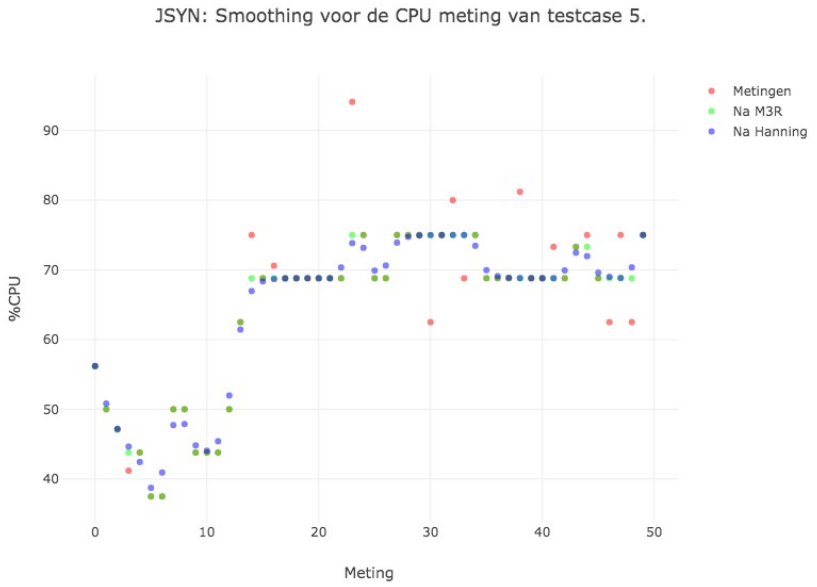
Figuur F.22: Smoothing van CPU testresultaten van testcase 2voor JSyn



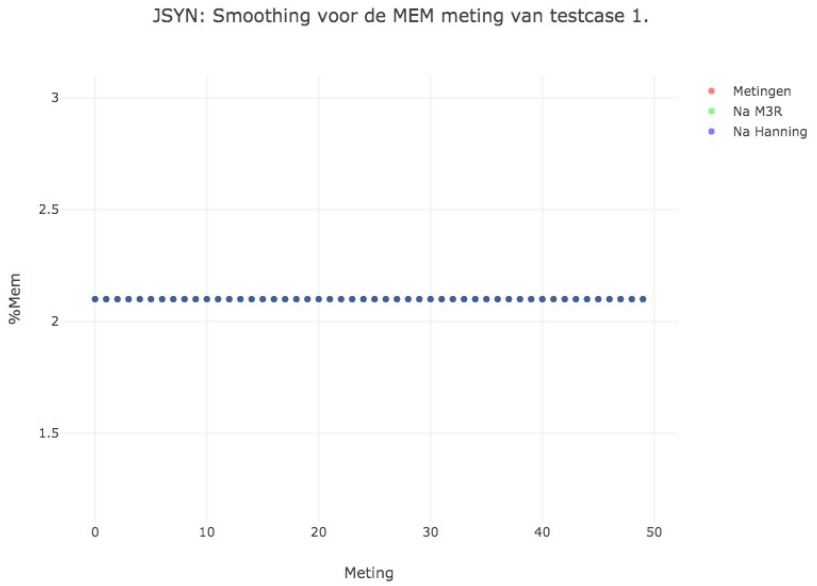
Figuur F.23: Smoothing van CPU testresultaten van testcase 3voor JSyn



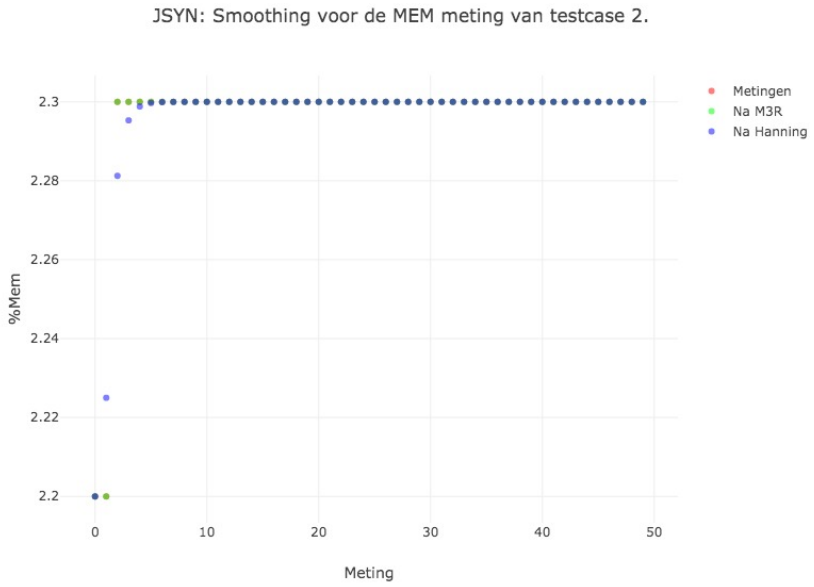
Figuur F.24: Smoothing van CPU testresultaten van testcase 4voor JSyn



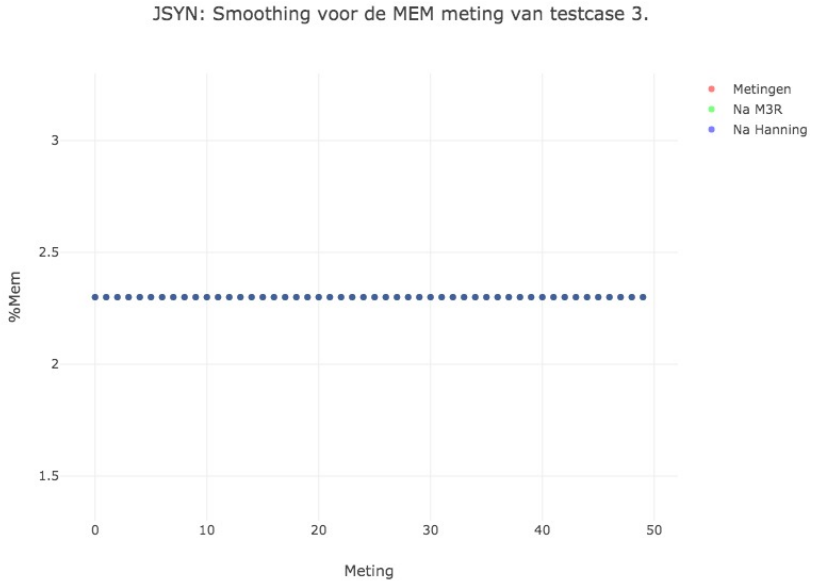
Figuur F.25: Smoothing van CPU testresultaten van testcase 5voor JSyn



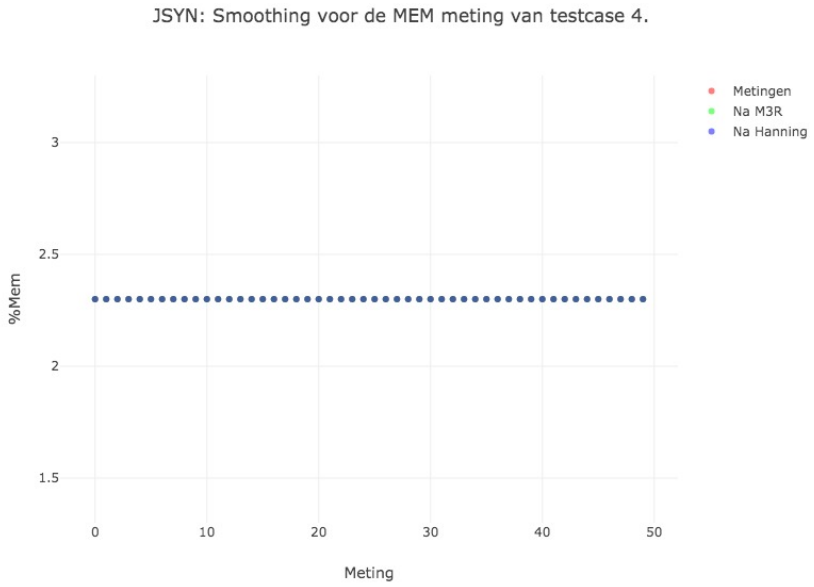
Figuur F.26: Smoothing van Mem testresultaten van testcase 1voor JSyn



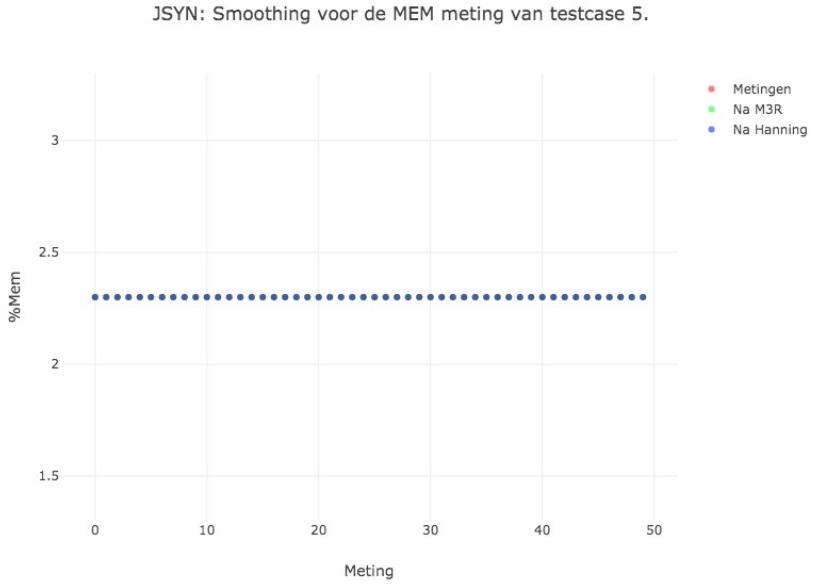
Figuur F.27: Smoothing van Mem testresultaten van testcase 2voor JSyn



Figuur F.28: Smoothing van Mem testresultaten van testcase 3voor JSyn



Figuur F.29: Smoothing van Mem testresultaten van testcase 4voor JSyn



Figuur F.30: Smoothing van Mem testresultaten van testcase 5voor JSyn