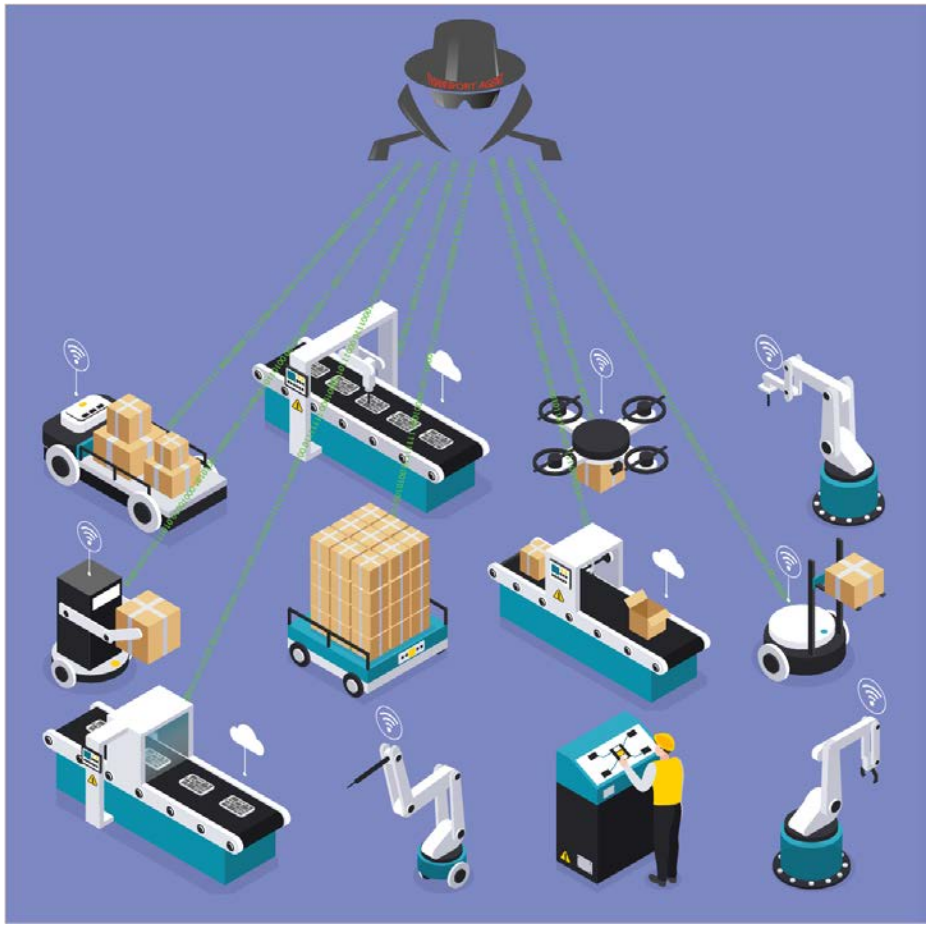
# Project Productiepadplanning 2.0 in een Agile Multiparallel Productiegrid

*Scriptie*



Joost Wagensveld-van Veen
1664713

Eerste Examinator:
Bas van den Berg

Stagebegeleider (tweede examinator):
Huib Aldewereld

Stagebedrijf:
Hogeschool Utrecht,
Leo van Moergestel

Datum: maandag 14 december 2020

# Samenvatting

Deze scriptie is geschreven als afstudeeropdracht van de opleiding Technische Informatica aan de Hogeschool Utrecht.

Om producten op kleine schaal te produceren, zal de huidige manier van produceren moeten veranderen door middel van een Agile aanpak. Het lectoraat Microsysteemtechnologie stelt dat dit het best gedaan kan worden met Equiplets: kleine herconfigureerbare productieplatforms. Deze zijn door middel van een grid met elkaar verbonden.

Doel van de afstudeeropdracht was om een simulatie te maken voor de beweging van de transportplatforms binnen een agent controlled productiegrid. Binnen deze opdracht is de hoofdvraag: Welke oplossingen zijn er om door middel van simulaties de efficiëntie van een Agile manufacturing productiegrid de productie te verbeteren ter validatie van de theorie? Om deze hoofdvraag te beantwoorden zijn deelvragen opgesteld: 1. Aan welke randvoorwaarden moet de simulatietool voldoen? 2. Welke simulatietool voldoet aan de randvoorwaarden gevonden in deelvraag 1 en is geschikt voor deze opdracht? 3. Wat zijn de beperkingen van de grid lay-out die zijn gebruikt in de paper (van Moergestel et al., 2018)? 4. Hoe kunnen deze beperkingen overkomen worden? 5. Welke invloed heeft het gebruik van een centrale padplanning agent op deze beperkingen? Om antwoorden te vinden op de gestelde vragen is literatuuronderzoek uitgevoerd en is een simulatie gemaakt.

Allereerst is er bepaald aan welke randvoorwaarden de simulatietool moet voldoen. Hierbij is gekeken naar de programmeertaal, het platform, de gebruikte algoritmen en het type simulatietool. Hieruit bleek dat RinSim de simulatietool leek die aan de randvoorwaarden voldeed en geschikt leek voor deze opdracht. Belangrijkste beperkingen van de grid lay-out waren de invloed van “wachtende” transportplatformen door het ontbreken van parkeerplekken en de manier waarop het grid wordt gebruikt. Tijdens de simulaties is gebleken dat het strak volgen van de lijnen van een grid niet het meest efficiënt is en voor complexe problemen kan zorgen. Op de vraag of centrale padplanning beter is dan decentrale is de conclusie dat decentrale padplanning beter is dan centrale padplanning als er sprake is van situaties met zowel middel tot hoge dynamiek, hoge urgentie en middel tot hoge schaal. In alle andere gevallen is decentrale padplanning dus wel efficiënter.

Concluderend kan worden gesteld dat centrale padplanning inderdaad efficiënter is, zolang er niet wordt voldaan aan de volgende drie criteria: 1.middel tot hoge dynamiek in het productiegrid; 2. hoge urgentie binnen het productiegrid; 3. middel tot hoge schaal binnen het productiegrid. Ook is de manier waarop het huidige grid is vormgegeven niet efficiënt omdat obstakels te veel invloed hebben of transportbewegingen.

De aanbeveling is om het transportgrid te gaan zien als open veld waarbinnen de transportplatformen zich vrij kunnen bewegen. Een open veld productie omgeving heeft mogelijk ook als voordeel dat rondom Equiplets waarvan bekend is dat deze drukker bezocht worden er ruimte vrijgehouden kan worden tot andere Equiplets.

Tijdens deze afstudeerperiode heb ik veel geleerd en veel plezier gehad met de padplanning algoritmen en agent controlled productie grids. Bij dit afstudeeronderzoek heb ik gemerkt dat ik een aantal zaken goed gingen, en ik een aantal struikelpunten had. Zo had ik moeite met het werken op afstand, de moeite die het kostte om contact te krijgen met mijn begeleiders, had ik problemen met mijn planning en trok ik te snel conclusies. Hierdoor heb ik de volgende leerdoelen op gesteld. Ik ben bij mijn toekomstige werkgever in staat op afstand te werken. Ik ben in staat op basis van wetenschappelijke bronnen een gewogen beslissing te nemen. Ik kan na drie maanden bij mijn toekomstige werkgever duidelijk communiceren naar mijn directe collega's. Ik kan aan het einde van mijn afstudeerperiode een reële planning maken van mijn werk waardoor ik de gestelde doelen haal.

# Afkortingen en acroniemen

| | |
|---|---|
| ABM | Agent Based Model (Agent-gebaseerd model) |
| AGV | Automated Guided Vehicle/Automatisch Geleide Voertuig |
| AI | Artificial Intelligence |
| BIM | Business IT & Management |
| GUI | Grafical User Interface |
| IDE | Integrated development environment |
| JSON | JavaScript Object Notation |
| MST | Lectoraat Microsysteemtechnologie |
| OS | Operating System |
| SDN | Software Defined Network(s) |
| XML | Extensible Markup Language |

# Inhoudsopgave

# 1. Inleiding

De opdracht die aan de basis staat van dit document is opgesteld als afsluiting van mijn opleiding Technische Informatica aan de Hogeschool Utrecht. Aanleiding voor deze opdracht was in eerste instantie dat er binnen de Hogeschool Utrecht een grote behoefte is aan een demonstrator die gemaakt is aan de hand van onderzoek wat jarenlang gedaan is binnen de Hogeschool Utrecht.

De Hogeschool Utrecht is voornamelijk bekend als opleidingsinstituut, er wordt echter bij de lectoraten ook veel onderzoek gedaan. Een van de producten uit deze onderzoeken vormt de grondslag voor de opdracht binnen deze afstudeerstage, namelijk de Equiplets. Een Equiplet is een autonome, modulaire, her-configureerbare, single-service, low-cost productiemachine (Telgen, 2017). De Equiplets zijn in meer of mindere mate het onderwerp geweest van diverse promotie-trajecten binnen de Hogeschool Utrecht.

Het concrete doel van deze stageopdracht was een Equiplet werkend krijgen die bijvoorbeeld op open dagen een demonstratie kan geven. Het vernieuwende hieraan was dat er momenteel geen werkende Equiplets zijn binnen de Hogeschool Utrecht.

Echter is er vanwege de coronacrisis besloten om de opdracht bij te stellen zodat deze geheel of gedeeltelijk vanuit huis gedaan kan worden. Dit betekent dat de opdracht meer op de theorie is gericht dan op hardware aangezien de hardware minder makkelijk vanuit huis te verkrijgen of te benaderen is. De opdracht is nu erop gericht om een door Leo van Moergestel geschreven artikel (van Moergestel et al., 2018) te valideren en aan te tonen dat de conclusies die gesteld worden kloppen. Het onderzoek beschreven in dit paper is gelieerd aan de equiplets, aangezien het gaat over het transport tussen de Equiplets. Deze beslissing is genomen toen de eerste versie van het plan van aanpak al in een definitieve fase zat, er is vervolgens een nieuw plan van aanpak geschreven (bijlage I).

In deze scriptie kunt u lezen over de aanleiding van de stage, de onderzoeksvraag, de onderzoekmethode, de resultaten en de opgeleverde producten.

De scriptie is als volgt opgebouwd. Eerst zal er uitgebreid ingegaan worden op de organisatorische context met daarin de uitleg wie de belangrijke partijen zijn binnen het afstudeerbedrijf. Dit is beschreven in hoofdstuk 2. Hoofdstuk 3 gaat uitgebreid in op de theoretische basis van de afstudeeropdracht. Dit hoofdstuk bevat onder andere kernbegrippen die binnen de afstudeeropdracht aan bod komen. In hoofdstuk 4 leest u over de onderzoeksvraag en de onderzoekmethode, in hoofdstuk 5 over de resultaten en de opgeleverde producten. Hoofdstuk 6 toont de Proof of Concept. Ten slotte vindt u in hoofdstuk 7 de evaluatie betreffende het proces van het afstuderen en het uitvoeren van het onderzoek.

# 2. Organisatorische context

Dit hoofdstuk beschrijft de organisatorische context van deze afstudeeropdracht, met de Hogeschool Utrecht als opdrachtgever.

*Hogeschool Utrecht* staat bekend als onderwijsinstituut, maar doet daarnaast ook veel praktijkgericht onderzoek (Hogeschool Utrecht, 2020a). Dit is onderzoek waarvan de vraagstelling wordt ingegeven door de beroepspraktijk en waarvan de opgedane kennis direct bij kan dragen aan die beroepspraktijk (Andriessen, 2014). Dit in tegenstelling tot fundamenteel wetenschappelijk onderzoek waarbij het doel is om bepaalde feiten of principes te ontdekken of te verifiëren.

De opdracht is direct gelinkt aan verschillenden onderzoeksprojecten zoals het promotieonderzoek van Daniel Telgen (Telgen, 2017), het promotieonderzoek (van Moergestel, 2014), maar de belangrijkste is uiteraard het paper van Leo van Moergestel *A Software Architecture for Transport in a Production Grid* (van Moergestel et al., 2018). Systemen waarmee ik gewerkt heb zijn software tools voor versiebeheer: Git; Java of Python als programmeeromgeving en in beperkte maten C/C++.

De opdracht is uitgevoerd binnen de onderzoeksgroep van het lectoraat Microsysteemtechnologie (MST) van de Hogeschool Utrecht (Hogeschool Utrecht, 2020b). Ik was verantwoordelijk voor het uitvoeren van de opdracht en alle zaken die nodig zijn geweest om de opdracht tot een goed resultaat te brengen. De dagelijkse begeleiding is gedaan door Leo van Moergestel. Ook is deze opdracht gesteund vanuit de Intelligent Systems Group van de Universiteit Utrecht (Intelligent Systems Group, 2015). Nini Salet heeft zich voornamelijk bezighouden met de procesbegeleiding. Ik heb de volgende taken uitgevoerd:

- Het verzamelen van al bestaande code en technieken en deze combineren en uitbreiden tot een werkende demonstrator inclusief de bijbehorende documentatie.
- Het opstellen van documenten die noodzakelijk zijn om de virtualisatie te reconstrueren.
- Het vergaren van benodigde informatie door middel van formele en informele interviews met Leo van Moergestel en andere experts.

De opdracht bouwt voort op het onderzoeksproject Agile Manufacturing waarin de Equiplets zijn ontworpen. De stageopdracht is direct gerelateerd aan het promotieonderzoek (van Moergestel, 2014) wat uitgevoerd is door Leo van Moergestel in de periode van 2010 tot en met 2014.

## 2.1 Bedrijfs-/persoonsgegevens

De volgende personen zijn betrokken bij de begeleiding van de student. In Bijlage I: Functioneel Design (CAFCR) zijn in hoofdstuk 2: CUSTOMER OBJECTIVES VIEW alle partijen weergegeven die op een of andere manier bij de afstudeeropdracht betrokken zijn. In tabel 1 zijn de contactgegevens verwerkt van de contactpersonen die vanuit het stagebedrijf zijn aangesteld als begeleiders.

*Tabel 1: Contactgegevens stagebedrijf*

| Naam | Functie | Rol binnen het project | Emailadres | Telefoonnummer |
|---|---|---|---|---|
| **Leo van Moergestel** | Hogeschool Hoofddocent/ Onderzoeker | Bedrijfsbegeleider | Leo.vanmoergestel@hu.nl | 06 839 522 27 |
| **Nini Salet** | Docent | Procesbegeleider | Nini.salet@hu.nl | 088 481 80 67 |

# 3. Theoretische basis

Dit hoofdstuk geeft de theoretische basis van de afstudeerscriptie. Het geeft een introductie op het onderzoek wat de aanleiding was voor deze scriptie, het geeft de lezer handvatten door middel van begrippen die gekend moeten worden, en het legt uit hoe een Agile manufacturing productiegrid werkt.

We leven in een wereld waar, op het gebied van productie, aanpassingen aan een productieproces zeer gewoon zijn geworden. Een klant wil steeds meer maatwerk en neemt minder genoegen met een universeel product. Dit betekent dat de huidige manier van produceren zal moeten veranderen door middel van een Agile aanpak. Het lectoraat Microsysteemtechnologie (Hogeschool Utrecht, 2020b) stelt dat deze veranderingen het best plaats kunnen vinden door middel van het gebruik van kleine herconfigureerbare productieplatforms genaamd Equiplets. Deze zijn door middel van een "grid" met elkaar verbonden (Puik & van Moergestel, 2010). Op deze manier is het mogelijk om zeer kleinschalig of zelfs losse individuele producten te fabriceren zonder daarvoor specifieke productielijnen op te zetten.

In het artikel *A Software Architecture for Transport in a Production Grid* (van Moergestel et al., 2018) wordt gesteld dat door alle afzonderlijke transportplatforms hun eigen agent de padplanning te laten doen bij een relatieve drukte tussen de 50% en 75% het grid verstopt raakt. Om dit te verhelpen en de algehele capaciteit van het productiegrid te verhogen is een oplossing bedacht. Deze oplossing stelt dat door de padplanning door een overkoepelende transportagent te laten doen het grid hogere aantallen transportplatforms aan kan. Dit is echter nog een theoretisch model en heeft dus onderbouwing nodig. Deze onderbouwing dient er te komen in de vorm van een simulatie (van Moergestel & Wagensveld, 2020c). Om deze simulatie te maken en te kunnen begrijpen moet men de volgende begrippen kennen:

## 3.1 Begrippen

***Agile Manufacturing:*** dit is een bepaalde manier van produceren om snel in te kunnen spelen op klantbehoeften en marktveranderingen, terwijl de kosten en kwaliteit nog steeds onder controle zijn. Het is erop gericht om maximale waarde voor de klant te realiseren met zo min mogelijk verspilling. Door verspillingen te elimineren gaan de operationele kosten omlaag, wat in het algemeen leidt tot een verbetering van het bedrijfsresultaat (Agile Methodologies for Industrial Production, 2020). Door middel van Agile Manufacturing kunnen producten gemaakt worden zoals de klant het wil, wanneer de klant het wil en tegen een kosten efficiënte prijs. Om dit daadwerkelijk te realiseren, zoals beschreven (Puik & van Moergestel, 2010) zullen agents de verschillende onderdelen van productie gaan regelen.

***Lean Manufacturing:*** dit wordt soms wel eens verward met Agile Manufacturing, echter is dit niet correct. Waar bij Agile juist gericht is op het zo intelligent mogelijk gebruiken van de (natuurlijke) hulpbronnen, wordt er bij Lean vooral gekeken naar hoe alle aspecten van de productie herzien kunnen worden en wordt alles verwijderd wat niet nodig is. Wat Lean en Agile gemeen hebben is dat ze beide sterk afhankelijk zijn van statistieken, prognoses en proactieve planning, waardoor kosten kunnen worden verlaagd tijdens, zeker als er minder geproduceerd wordt (Redwood Logistics, 2020).

***Equiplets:*** (Telgen, 2017; van Moergestel, 2014) Een methode voor Agile Manufacturing is om kleine hoeveelheden of zelfs unieke producten tegen een lage kostprijs te fabriceren. Om dit te verwezenlijken zijn voor de onderzoeken van Daniel Telgen en Leo van Moergestel speciale goedkope productieplatforms ontwikkeld. Deze herconfigureerbare productiemachines noemen we Equiplets (van Moergestel, 2014).

***Transportplatforms:*** Het transport van de producten van Equiplet naar Equiplet is heel anders dan bij standaardproductie. Elk product kan zijn eigen unieke pad langs de apparaten hebben. Om producten tussen de verschillende Equiplets te transporteren zijn geautomatiseerde transportplatforms nodig (van Moergestel, 2014).

***Agent:*** (Wooldridge & Jennings, 1995) het gaat hier om een software agent en dit is een embedded computersysteem dat zich in een bepaalde omgeving bevindt. De agent is in staat tot het ondernemen van flexibele autonome acties in deze omgeving om zijn ontwerpdoelstellingen te bereiken. Agents zijn duidelijk herkenbare entiteiten die zelfstandig problemen proberen op te lossen, met strak gedefinieerde grenzen en interfaces (van Moergestel, 2014).

***Agent controlled productiegrid:*** Dit is een verzameling van Equiplets in een gridopstelling geplaatst en gekoppeld met een snelle netwerkverbinding, hierdoor is het mogelijk om een aantal verschillende producten tegelijk te produceren. Om dit grid echt Agile te laten werken is er onderliggend een agent gebaseerd software nodig voor de aansturing. Hierdoor is het mogelijk tot het toepassen van flexibele parallelle productie (Telgen, 2017; van Moergestel, 2014).

***Software Defined Network:*** dit is een manier van het managen van een netwerk waarbij de switches naar een centrale controller communiceren. Deze berekent op zijn beurt de beste weg via waar de datapakketten worden verstuurd (Kurose & Ross, 2017). Dit in tegenstelling tot de werkwijze waarbij de switches met elkaar communiceren om zelf de kortste/snelste weg te berekenen. Dit concept is een goede oplossing voor het vereenvoudigen van het agent gestuurde transportplatform. Natuurlijk is een transportplatform geen router, maar er zijn genoeg overeenkomsten (van Moergestel et al., 2018). Ook zijn er theorieën die stellen dat in geval van gemiddeld tot hoog dynamische scenario's, met hoge urgentie en middelgrote tot grote schaal centrale algoritmen minder efficiënt zijn dan decentrale algoritmen (R. R. van Lon & Holvoet, 2017).

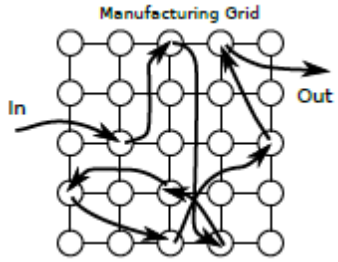
***Agent-based model:*** in het Nederlands: agent gebaseerde simulatie, is een simulatie waarbij een deel van de werkelijkheid wordt nabootst met een model waarbij het effect van de actie en interactie van de afzonderlijke onderdelen op het systeem als geheel wordt bestudeerd (Railsback & Grimm, 2019).

## 3.2 Werking Agile manufacturing productiegrid

De productiestappen binnen het productiegrid worden gedaan door de Equiplets. Iedere Equiplet is in staat één of meerdere productiestappen uit te voeren. Hierbij bij kan een productiestap zijn opgedeeld in groep van gecoördineerde of samenhangende acties op een product. Na iedere productiestap is de toestand van het product stabiel, dit betekent dat het product getransporteerd kan worden of tijdelijk opgeslagen tussen twee stappen.

Om een product te maken moet een reeks van deze productiestappen worden uitgevoerd. Als je dit wilt bereiken, moet een set Equiplets in een bepaalde volgorde worden gebruikt. Om het product (en de bijbehorende componenten) te verplaatsen tussen de Equiplets wordt gebruik gemaakt van een transportplatform. Het transportplatform wordt voordat deze het grid ingaat eerst geladen met componenten die nodig zijn voor de productie. Vervolgens zal het transportplatform het grid ingaan waar de componenten door de Equiplets worden gebruikt om een product (of halffabricaat) te creëren. Als de Equiplet klaar is zal het transportplatform via het grid naar de volgende Equiplet gaan voor de volgende stappen in het productieproces om het eindproduct te maken.

Verschillende producten hebben specifieke productstappen nodig in hun eigen specifieke volgorde. De kracht en veelzijdigheid van dit systeem is dat elk product zijn eigen route in het grid kan hebben, wat kan resulteren in een uniek product. In figuur 1 is een voorbeeld te zien van een route die een product zou kunnen volgen door het grid. Door het maken van halffabricaten kunnen ook complexere producten volgens hetzelfde principe worden gebouwd.




*Figuur 1: Route van bepaald product (Leo van Moergestel et al., 2014)*

De productiestappen komen als volgt tot stand: als start bij het maken van een product wordt een unieke productagent gegenereerd. Deze agent weet welke productiestappen er genomen moeten worden en welke componenten gebruikt moeten worden om een product te vervaardigen. De productagent krijgt een transportsysteem toegewezen en die haalt op zijn beurt de componenten op. De productagent zoekt op een blackboard systeem op welke Equiplets nodig zijn om de productiestappen uit te voeren. Op dit blackboard systeem wordt bijgehouden welke stappen door welke Equiplet uitgevoerd kunnen worden.

Voordat een Equiplet definitief wordt gekozen vraagt de productagent eerst na bij de desbetreffende Equiplet echt in staat is om de specifieke productiestap uit te voeren, gegeven de specifieke parameters. De Equiplet zal hiervoor zelf een nauwkeurige simulatie van de gevraagde productiestap uitvoeren met die parameters. Als resultaat geeft de Equiplet de mogelijkheid en de tijd terug die nodig is om deze productiestap te voltooien. Op deze manier werkt de productagent alle productiestappen af om een lijst te genereren met Equiplets die aangedaan moeten worden en in welke volgorde.

# 4. De opdracht

In dit hoofdstuk leest u meer over de opdracht die ten grondslag van deze scriptie ligt. Een beschrijving van de opdracht, uitleg over wat de opdrachtgever wil, de aanpak van de opdracht en de te gebruiken onderzoeksmethoden, de hoofdvraag en deelvragen die beantwoord moeten worden.

## 4.1. Beschrijving van de opdracht

Het doel van de opdracht is om een simulatie te maken voor de beweging van de transportplatforms binnen een agent controlled productiegrid. Dit is om aan te tonen dat het aansturen van de transportplatforms door middel van een aansturende transport-plannings-agent sneller/beter is voor het efficiënt berekenen van routes dan wanneer alle afzonderlijke transportplatforms hun eigen (zelf plannende) transportagent hebben. Een hogere efficiëntie wordt vooral behaald door minder opstoppingen waardoor een product binnen een kortere tijd klaar is.

Binnen het promotieonderzoek van Leo van Moergestel (van Moergestel, 2014) zijn diverse simulaties ontworpen om te demonstreren hoe het systeem functioneert. Het eerste gedeelte van de opdracht is geweest: het onderzoeken van de simulaties die al gedaan zijn voor de situatie met de transportagent in het transportplatform. Tijdens dit onderzoek is nadrukkelijk gekeken welke tekortkomingen er allemaal zijn en hoe die opgelost kunnen worden om een hogere productiviteit binnen het productiegrid te verwezenlijken. Een van de tekortkomingen binnen de eerdere simulaties was onder andere dat als een Equiplet “bezet” was deze niet gepasseerd kon worden en er dus filevorming kon ontstaan.

Omdat het compleet van de grond af opbouwen van een simulatie veel tijd kost om te ontwikkelen en te testen is er besloten om te kijken welke bestaande onderdelen er al zijn. In gesprekken met verschillende mensen van binnen en buiten Hogeschool Utrecht kwam naar voren dat het goed zou zijn om te kijken naar aan aantal verschillende bestaande oplossingen voor het simulatievraagstuk. Het onderzoeken van bestaande simulatie softwarepakketten is ook geadviseerd in overleg met Huib Aldewereld (Aldewereld, 2020). In hoofdstuk 5.1 is te lezen welke simulatiepakketten (simulatietools) zijn bekeken. Aan het begin van de opdracht is de volgende lijst opgesteld:

- RePast Simphony
- MESA
- Simulink
- Analytic Solver Simulation
- RinSim

Er zijn tijdens het literatuuronderzoek nog wel andere simulatiepakketten langsgekomen, echter zijn deze niet beschreven in deze scriptie omdat deze (op eerste gezicht) niet goed binnen de scope van de opdracht pasten.

De simulatie zal zoals uitgelegd in de theoretische basis (hoofdstuk 3) gevoed worden door informatie vanuit een XML-bestand. Er is ook nog gekeken naar andere internet compatible formats zoals bijvoorbeeld JSON, echter leverde dit geen extra functionaliteit en is dit dus niet geïmplementeerd.

Tijdens deze stage is onderzoek gedaan naar de simulaties die zijn gemaakt voor het transportsysteem waarbij de transportplatforms zelfstandig de routeplanning doen. Ook is er onderzoek gedaan naar hoe het transport zou verlopen als er gebruik gemaakt zou worden van de “traditionele” transportmethode, zoals bijvoorbeeld transportbanden. De verschillende transportmethoden moeten namelijk wel goed met elkaar vergeleken kunnen worden. Na dit grootschalige literatuuronderzoek is een simulatie gemaakt van een agent controlled productiegrid. Deze simulatie kent een aantal varianten. Er is gevarieerd in de architectuur van het grid en er is gevarieerd in de aansturing/padplanning van de transportplatformen. Daarbij is een decentrale padplanning vergeleken met een centrale padplanning, vergelijkbaar met de Central controller in een Software Defined Network (SDN).

Er is een iteratieve ontwikkelmethode gehanteerd, waar een aantal werkende systemen in worden opgeleverd met oplopende complexiteit.

In het eindproduct zijn de volgende dingen verwezenlijkt:

- Een simulatie van een agent controlled productiegrid;
- In deze simulatie is het mogelijk om op basis van verschillende XML-beschrijvingen van productie de mogelijke oplossingen voor een productiepad in een productiegrid te genereren;
- De simulatie zal per product, een productiepadplanning berekenen inclusief tijd die nodig is om dit product te vervaardigen.

Voor het maken van de software is voornamelijk gewerkt met JAVA. Dit omdat met de keuze voor RinSim het gebruik van JAVA het meest voor de hand lag. Leo van Moergestel heeft herhaaldelijk kenbaar gemaakt dat hij een lichte voorkeur had voor JAVA of C/C++ als hij het zou moeten maken. Omdat er geen eerder gemaakte simulaties voor handen waren was er dus ook geen programmeertaal gedefinieerd. Echter voordat ik echt met literatuuronderzoek ben begonnen dacht ik met behulp van Python snel en eenvoudig een concept in software te kunnen creëren. Dit is dan ook de reden dat ik een deel van mijn producten in Python heb gemaakt. Dit resulteerde in het uiteindelijke gebruik van twee programmeertalen binnen deze afstudeeropdracht.

Er is gekozen om binnen deze opdracht te werken met XML-bestanden (van Moergestel & Wagensveld, 2020b). Hiervoor is gekozen omdat deze makkelijk gemaakt kunnen worden vanuit een web interface waar de klant producten kan “bestellen” (van Moergestel & Wagensveld, 2020b). Bijkomend voordeel is dat XML een bestandformaat is dat zowel door mensen als door machines makkelijk te lezen is. Dit XML-bestand zal informatie bevatten over de hoeveelheid transportplatforms die rond zullen rijden, wanneer deze het grid zullen binnen gaan, wat het product is wat gemaakt moet worden en welke stappen ze daarvoor moeten ondergaan. Ook zal er een XML-bestand zijn die het productiegrid zal beschrijven met daarin onder andere het aantal Equiplets in het grid, hoe de paden lopen, waar de in- en uitgangen zitten in het grid en wat de eigenschappen zijn van de verschillende Equiplets

## 4.2. Wat de opdrachtgever wil

Omdat er geen hardware of software platform beschikbaar is van een agile manufacturing productie grid wil de opdrachtgever als eindresultaat een werkende simulatie die op basis van een of meerdere XML-inputs de simulatie uitvoert en productiepaden genereert. Ook zal er een voorstel gedaan moeten worden hoe de grid architectuur eruit dient te zien om de efficiëntie van het grid te verbeteren.

Verder wil de opdrachtgever expliciet dat er geen tijd gestoken wordt in grafische toeters of bellen. Hiermee wordt bedoeld dat er geen scherm gemaakt moet worden waarop de karretjes rond zullen rijden. Maar er zal wel een grafische output zijn die als mens te interpreteren is. De output mag zelfs in een XML-bestand zijn dat tijdstippen en paden van het productieproces toont. Deze output zal dan door middel van een script of een spreadsheetprogramma (bijvoorbeeld Excel) een grafiek of diagram genereren. Door met behulp van test sets met een voorspelbare uitkomst in de simulatie te draaien en te controleren of het systeem dat ook genereert kan de werking van het systeem gevalideerd worden (van Moergestel & Wagensveld, 2020b).

Doel van de opdracht is het verkrijgen van goede vergelijkingsresultaten om aan te tonen dat een overkoepelende transportagent de oplossing is voor capaciteitsproblemen in het transport binnen een productiegrid.

Het product dat opgeleverd zal worden is een simulatie, die voornamelijk resultaten zal tonen en niet een zeer geavanceerde grafische visualisatie. Ook zal het een advies bevatten om tekortkomingen aan een productiegrid te beperken. Binnen dit project zal ik voornamelijk de software schrijven en zorgen voor een gedetailleerde uitleg over wat er in de simulatie gebeurt alsmede een wetenschappelijke onderbouwing daarvan.

## 4.3. Aanpak van de opdracht

Uit simulaties die in eerdere projecten zijn gemaakt zijn resultaten beschikbaar die als vergelijking zullen dienen voor de simulatie die gemaakt zal worden binnen deze afstudeeropdracht. Hierdoor zal op basis van een globale aanpak een agile plan moeten worden uitgevoerd. De belangrijkste module waar mee gewerkt zal moeten worden is de simulatie van de transportplatforms.

De globale volgorde in de planning zal zijn:

- Functioneel design maken, met de CAFCR methodiek als basis;
- Uitzoeken welke simulaties er eerder gemaakt zijn;
- Een productiegrid volgens het ontwerp (van Moergestel, 2014) maken;
- Productsimulatie maken;
- 1 product door het ontworpen productiegrid laten produceren;
- Onderzoek doen naar hoeveel het grid nu aan kan.

De simulatie moet aantonen dat een grid efficiënter is als de padplanning centraal gedaan wordt en niet decentraal. Als je dit vertaalt naar de praktijk zal het zo gaan: Een transportagent vraagt aan een planningsagent ‘hoe kom ik het snelste/beste van mijn huidige locatie naar locatie X’. Dit zal beter gaan dan als de transportagent zelf gaat berekenen hoe dit het beste kan. Binnen deze opdracht zal ik dit met een simulatie onderzoeken.

## 4.4. Onderzoeksmethoden

Onderzoeksmethoden zijn vooral literatuuronderzoek geweest. Eerst is vooral onderzoek gedaan naar de door Leo van Moergestel aangeraden vakliteratuur (Telgen, 2017; Telgen, Puik, van Moergestel, Bakker, & Meyer, 2015; van Moergestel, 2014; van Moergestel et al., 2018; van Moergestel, Telgen, Puik, & Meyer, 2014). Daarna is er vooral onderzoek gedaan naar bestaande simulatietools, hierbij is Google Scholar (Google Scholar, 2020) onmisbaar gebleken. Ook is er via teams, whatsapp en telefoongesprekken nauw contact gehouden met Leo van Moergestel in verband met zijn kennis. Vervolgens zijn beschikbare en opgedane kennis, software, ideeën en simulaties uitgewerkt. Daarna zijn alle ontwikkelingen en experimenten gecombineerd om de simulatie te realiseren.

Tijdens de stageopdracht heb ik mij voornamelijk ingelezen op de diverse papers die door Leo van Moergestel en andere leden binnen het onderzoek collectief zijn geschreven en daarbij heb ik vooral gericht gezocht naar papers over de diverse transportmethoden (Telgen, 2017; van Moergestel et al., 2018; LJM van Moergestel et al., 2014). Uiteraard is een belangrijk naslagwerk binnen deze afstudeeropdracht geweest het proefschrift van Leo van Moergestel (van Moergestel, 2014). Buiten het onderzoek collectief waarbinnen Leo van Moergestel actief is zijn voornamelijk de onderzoekers Nicholas R. Jennings en Stefan Bussmann die ook veel onderzoek hebben gedaan naar Agent-based control systems (Jennings & Bussmann, 2003). Ook is er uitgebreid kennis opgedaan over RinSim en overige onderzoeken die gedaan zijn (R. van Lon, 2018b; R. van Lon & Holvoet, 2012; R. R. van Lon & Holvoet, 2017).

Verder is er via Google Scholar gezocht op de volgende zoektermen of combinaties van deze zoektermen met diverse bovengenoemde auteurs:

- Simulation
- Simulation software
- Agent technology
- Production grid
- Agile manufacturing
- Lean manufacturing
- Software Defined Network
- RinSim

Dit zijn ook trefwoorden waarmee de opdracht wordt gekarakteriseerd.

Bij het aanvangen van de afstudeeropdracht is de bedoeling geweest de eerdere simulaties te bekijken, echter zijn die nergens meer te vinden. Het enige wat nog beschikbaar is geweest zijn het artikel (van Moergestel et al., 2018) alsmede het proefschrift (van Moergestel, 2014), deze zijn extra goed bestudeerd. Ook zijn de andere projecten die raakvlakken hebben met dit project bestudeerd.

Tijdens het hele afstudeerproject is geprobeerd een agile aanpak te hanteren om de simulatie te maken waarop de verschillende strategieën uitgeprobeerd konden worden om stagnatie in het productiegrid aan te pakken.

## 4.5. Hoofdvraag

Het hogere doel is het demonstreren dat een productiegrid met een overkoepelende transportagent beter werkt dan een productiegrid waarbij de transportagent in de transportplatforms de routing berekent.

Binnen deze opdracht is de hoofdvraag: Welke oplossingen zijn er om door middel van simulaties de efficiëntie van een Agile manufacturing productiegrid de productie te verbeteren ter validatie van de theorie (van Moergestel et al., 2018)?

De oplossingen zijn gezocht in de architectuur van het productiegrid, de manier van padplanning, maar ook in de stappen die ondernomen moeten worden als een deadlock situatie dreigt te ontstaan/is ontstaan.

## 4.6. Deelvragen

Gaandeweg de afstudeeropdracht zijn om de hoofdvraag te beantwoorden de volgende deelvragen geformuleerd en onderzocht.

1. Aan welke randvoorwaarden moet de simulatietool voldoen?
2. Welke simulatietool voldoet aan de randvoorwaarden gevonden in deelvraag 1 en is geschikt voor deze opdracht?
3. Wat zijn de beperkingen van de grid lay-out die zijn gebruikt in de paper (van Moergestel et al., 2018)?
4. Hoe kunnen deze beperkingen overkomen worden?
5. Welke invloed heeft het gebruik van een centrale padplanning agent op deze beperkingen?

In het volgende hoofdstuk zullen deze vragen uitgebreid beantwoord worden.

# 5. Uitwerking van de opdracht

Deze afstudeeropdracht heeft de agile methodiek gevolgd waarbij er in korte sprints is gewerkt. Hierbij zijn de verschillende deelvragen bepaald door het te behalen resultaat aan het einde van de verschillende sprints. In dit hoofdstuk zal ik deze deelvragen stuk voor stuk beantwoorden.

## 5.1 Randvoorwaarden simulatietool

De eerste deelvraag is aan welke randvoorwaarden de simulatietool moet voldoen. Om zo snel mogelijk aan de slag te kunnen is in de eerste sprint onderzoek gedaan naar welke simulatietool het beste gebruikt kan worden binnen deze opdracht. Hierbij zijn de volgende eisen opgesteld. Bij voorkeur moet het een gratis open-source simulatietool zijn waarbij centrale en decentrale planningsalgoritmen getest kunnen worden op een gesimuleerd productiegrid. Het moet een niet al te hoge leercurve hebben aangezien de totale tijd waarbinnen het volledige project plaatsvindt enigszins beperkt is en het behalen van resultaten prioriteit heeft. Deze eisen heb ik gekregen van de opdrachtgever, en zijn in verschillende gesprekken met de opdrachtgever afgestemd.

Het moet eenvoudig zijn om diverse logistieke scenario's op te zetten zoals bijvoorbeeld de richtingsafhankelijkheid van paden in het grid en hoe het grid verbonden is. Als een algoritme gekozen moet worden zal het Dijkstra algoritme (Sniedovich, 2006) gebruikt worden. Er zal echter niet de mogelijkheid zijn tot het kiezen van meerdere padplanning s algoritmen, omdat dit voor extra complexiteit zorgt.

Met de opdrachtgever is ook afgestemd dat de simulatietool programmeerbaar is, zodat in de simulatie gevonden oplossingen hergebruikt kunnen worden, eventueel aangepast aan toekomstige simulaties of de praktijk, door ze waar nodig te herprogrammeren.

Voor wat betreft de programmeertaal heb ik de voorwaarde gesteld dat het een programmeertaal is die in mijn curriculum aan bod is gekomen. Dit zijn voornamelijk Java, C/C++ en Python, verder zijn er geen specifieke voorwaarden. Leo van Moergestel heeft de voorkeur uitgesproken voor Java, maar ziet dit zeker niet als voorwaarde (van Moergestel & Wagensveld, 2020a). Wel dient er geprogrammeerd te worden, dit moet dus mogelijk zijn in de simulatietool. Ook moet de simulatietool platformonafhankelijk zijn, dit moet minimaal beschikbaar zijn op Windows en Linux. Dit is een wens van de opdrachtgever, omdat Windows het basisplatform is wat vanuit school gebruikt wordt en Linux omdat dit gebruikt is voor het ontwikkelen van de Equiplets. Zo zou het in de toekomst mogelijk moeten kunnen zijn deze simulatie en software van de Equiplets met elkaar te combineren.

Het is een pre als de simulatietool door middel van tekstbestanden (bijvoorbeeld XML) van informatie voorzien kan worden, is dit niet het geval dan kan deze functionaliteit ook later ingevoerd worden.

Het antwoord op de vraag 'aan welke randvoorwaarden moet de simulatietool voldoen?' is terug te vinden in tabel 2.

Bij hoofdstuk 5.2 zal ik deze tabel vullen met informatie over de verschillende simulatietools.

*Tabel 2: Beslismatrix*

| | Programmeertaal | | | | Platform | | | | | | Algoritmen | | | Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Naam | C/C++ | Java | Python | Overige | Windows | Linux | macOS | Leercurve (0-7) | Open-Source | Betaald | Centraal | Decentraal | Keuze uit meerdere | Verkeer | Systeem | Logistiek | Sociale problemen |
| | | | | | | | | | | | | | | | | | |

## 5.2 Keuze simulatietool

In deze sprint was het doel de vraag 'Welke simulatietool voldoet aan de randvoorwaarden gevonden in deelvraag 1 en is geschikt voor deze opdracht?' te beantwoorden. Omdat het binnen dit project om een simulatie gaat waarbij de resultaten betrouwbaar moeten zijn en goed vergelijkbaar moeten zijn is er besloten om niet heel de simulatie zelf te programmeren. Wat ook heeft meegespeeld is dat het volledig zelf maken van een simulatie veel tijd kost om goed te ontwikkelen en te testen. Bovendien heeft dit een hoog "opnieuw het wiel uitvinden" gehalte, wat ook wel bleek tijdens het uitwerken van het Functioneel document (bijgevoegd in Bijlage I: Functioneel Design (CAFCR)). Dit document omvat ook de basis van deze paragraaf 5.2. Er zal in deze sectie gekeken worden naar een aantal verschillende bestaande simulatietools en er zal een worden gekozen. De beslissing om een bestaande simulatietool te kiezen boven het volledig zelf ontwerpen en maken worden ook bekrachtigd door het advies vanuit de Hogeschool Utrecht (Aldewereld, 2020), enkele van de onderstaande tools zijn onder ander afkomstig uit dat advies.

De volgende simulatietools zijn bekeken:

- RePast Simphony
- Simulink
- Analytic Solver Simulation
- RinSim
- NetLogo
- MESA

Er zijn nog meer simulatietools buiten de hier genoemde. De reden dat die niet zijn meegenomen in deze vergelijking is meerledig: sommigen vielen buiten de scope, sommigen waren niet of slecht gedocumenteerd of anderen waren al lang niet meer bijgewerkt. Als een simulatietool zich vooral lijkt te focussen op het simuleren van (psycho-)sociale problemen was dat voor mij ook een reden om niet verder te kijken.

### 5.2.1 RePast Simphony

Repast Simphony (North et al., 2013) is een op agenttechnologie gebaseerde modellering toolkit en is een platformonafhankelijk Java-gebaseerd simulatiesysteem. Repast ondersteunt de ontwikkeling van uiterst flexibele modellen van interacterende agents voor gebruik op krachtige computers en computerclusters. Repast Simphony-modellen kunnen in verschillende vormen worden ontwikkeld, het ReLogo-dialect van Logo (Abelson, Goodman, & Rudolph, 1974; Ozik, Collier, Murphy, & North, 2013). De structuur is modulair opgezet zodat modules aangepast of vervangen kunnen worden. Er zijn ook modules die het mogelijk maken Java, C#, C++, Visual Basic.Net, Prolog en Python te gebruiken, die allemaal in elkaar kunnen worden verweven. De variant van Repast die gemaakt is voor supercomputers maakt het ook mogelijk C++ te gebruiken in meer of mindere maten. RePast is vooral gericht op Rapid prototyping waarbij de lage leercurve belangrijker is en een degelijke programmeeromgeving (IDE) minder van belang lijkt te zijn. Wat betreft de leercurve, die is heel laag als alleen de kern (core) modules wordt gebruikt, dit is de ReLogo variant van RePast. Als echter van andere programmeertalen gebruik gemaakt wordt gaat deze al snel flink omhoog, daarom is in de beslismatrix de leercurve van 2 tot 6. Bovendien is Repast gespecialiseerd in het simuleren van sociale problemen.

### 5.2.2 Simulink

Simulink (The MathWorks Inc, 2020b) is een betaalde simulatietool die is gemaakt om een systeem te ontwerpen en te simuleren voordat overgegaan wordt op hardware. Het geeft de mogelijkheid ontwerpen te verkennen en te implementeren die anders misschien niet zou worden overwegen, zonder C, C++ of andere “hardware description language” te hoeven schrijven. Alle door Simulink gegenereerde code kan echter wel aangepast worden. Omdat dit niet echt het idee is binnen dit project valt deze om die reden al af, en zal voor dit project niet dieper in deze tool gedoken worden. Het voordeel van Simulink is wel dat het heel goed gecombineerd kan worden met MatLab (The MathWorks Inc, 2020a), wat de mogelijkheden erg vergroot.

### 5.2.3 Analytic Solver Simulation

Analytic Solver Simulation (Frontline Systems Inc, 2020) is een solver. Solvers zijn softwaretools die helpen door middel van een mathematische simulatie de beste manier te vinden om schaarse middelen toe te wijzen. De middelen kunnen grondstoffen, machinetijd of manuren, geld of iets anders (in beperkte voorraad) zijn. Dit past dus heel goed in de ideologie van Lean en Agile produceren. De "beste" of optimale oplossing kan het maximaliseren van de winst, het minimaliseren van de kosten of het bereiken van de best mogelijke kwaliteit betekenen. Frontline systems maak in zijn solvers gebruik van het Monte-Carlo simulatie principe. Dit zijn simulaties die door middel van het herhalen van dezelfde simulatie met willekeurige startwaarden probeert statistisch tot een eindresultaat te komen (Raychaudhuri, 2008). Het is mij niet bekend of deze simulatie gebruik maakt van centrale of decentrale algoritmen. Op deze manier kan een bijna oneindige verscheidenheid aan problemen worden aangepakt. Deze Analytic Solver Simulation werkt als extension op Excel van Microsoft en zelf beweren ze honderd keer sneller resultaten te genereren uit een simulatie dan hun concurrenten. Frontline Systems heeft ook een SDK ontwikkeld die gekoppeld kan worden, maar aangezien verschillende onderdelen wel heel ver afwijken van alle voorwaarden die gesteld zijn valt deze tool ook af als oplossing binnen deze opdracht. Bovendien is de prijs van $1,995.00 een goed argument om verder te zoeken.

### 5.2.4 RinSim

RinSim (R. van Lon, 2018b) is een uitbreidbare simulator die specifiek is gebouwd voor logistieke simulaties met ondersteuning voor gecentraliseerde en gedecentraliseerde algoritmen voor logistieke problemen (pickup and return) en (Automatisch geleide voertuigen) AGV-routing. De simulator is volgens de omschrijving gericht op eenvoud en consistentie, waardoor het ideaal is voor het uitvoeren van wetenschappelijke simulaties. Verder heeft bij het ontwerpen van de software de softwarekwaliteit bovenaan gestaan, wat resulteert in een steeds beter wordende testsuite en documentatie. Wat het ook een erg interessant softwarepakket maakt is dat het specifiek is ontwikkeld voor wetenschappelijk onderzoek en het daarvoor ook wordt gebruikt binnen de Distrinet Research Group van de KU Leuven (R. van Lon & Holvoet, 2012). Een groot voordeel is, is dat RinSim goed geïntegreerd kan worden in IntelliJ IDEA (JetBrains s.r.o, 2020), wat een zeer uitgebreide IDE is voor Java. Wat een nadeel is is dat RinSim echter wel een kleine gebruikerscommunity heeft buiten de KU Leuven. Dit ontdekte ik in een laat stadium van het project.

### 5.2.5 NetLogo

NetLogo (Wilensky, 2016) is een multi-agent programmeerbare simulatieomgeving. Ook dit platform wordt door heel veel studenten, docenten en onderzoekers over de hele wereld gebruikt. Het is geschreven in Scala en Java en maakt gebruik van de programmeertaal LOGO (Abelson et al., 1974; Wilensky, 2016). Daarom klinkt het in mijn ogen minder geschikt voor dit project. NetLogo lijkt zich voornamelijk te richten op simulaties van (psycho-)sociale problemen. Bij NetLogo lijkt net als bij RePast de leercurve belangrijker dan de uitgebreide programmeeromgeving. Wat wel interessant kan zijn is dat NetLogo gebruikt kan worden als extension binnen een groot aantal omgevingen, waaronder Python, Arduino, maar ook MIDI (geluid) en vele anderen (Wilensky, 2019).

### 5.2.6 MESA

Ook MESA (Kazil et al., 2020) is een agent-gebaseerd modelleer toolkit, echter dan in Python. Het stelt gebruikers in staat snel agent-gebaseerde modellen te creëren met behulp van ingebouwde kerncomponenten (zoals ruimtelijke grids en agentplanners) of aangepaste implementaties. Deze zijn te visualiseren met behulp van een (internet)browser-interface en de resultaten zijn te analyseren met behulp van de gegevensanalysetools van Python. MESA is gemaakt om op basis van Python 3 een tegenhanger te zijn van NetLogo en Repast. MESA wordt veelal gebruikt om simulaties te maken van sociale, economische en ruimtelijke processen, het is daarom minder geschikt voor logistieke problemen (Dean et al., 2000). Ik heb niets kunnen vinden over hoe de algoritmen geïmplementeerd worden in MESA en of deze centraal of decentraal geregeld worden.

### 5.2.7 Beslismatrix

Tabel 2 laat de beslismatrix zien, met per tool de eigenschappen, zoals de programmeertaal, het OS waarop de tool werkt, het algoritme en wat voor type simulatie. Het kenmerk “Keuze van meerdere” bij algoritmen is bij vrijwel alle simulatietools aangevinkt, dit wil echter niet zeggen dat dit voor alle tools het aanpassen van een padplannings algoritmen eenvoudig is. Bij sommige tools is dit zeer diep in de soft verwerkt.

*Tabel 3 Beslismatrix simulatietools*

| | Programmeertaal | | | | Platform | | | | | | Algoritmen | | | Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Naam | C/C++ | Java | Python | Overige | Windows | Linux | macOS | Leercurve (0-7) | Open-Source | Betaald | Centraal | Decentraal | Keuze uit meerdere | Verkeer | Systeem | Logistiek | Sociale problemen |
| RePast Simphony | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2-6 | ✓ | | | ✓ | ✓ | | | | ✓ |
| Simulink | | | | ✓ | ✓ | ✓ | | 6 | | ✓ | ✓ | | ✓ | | ✓ | | |
| Analytic Solver Simulation | | | | ✓ | ✓ | ✓ | ✓ | 5 | | ✓ | | | | ✓ | ✓ | ✓ | |
| RinSim | | ✓ | | | ✓ | ✓ | ✓ | 3 | ✓ | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| NetLogo | | ✓[1] | ✓[1] | ✓ | ✓ | | ✓ | 2-3 | ✓ | | | ✓ | ✓ | ✓ | ✓ | | ✓ |
| MESA | | | ✓ | | ✓ | ✓ | ✓ | 4 | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ |

### 5.2.8 Onderbouwing keuze simulatietool

In deze sectie is de vraag Welke simulatietool voldoet aan de randvoorwaarden gevonden in deelvraag 1 en is geschikt voor deze opdracht?' uitgezocht. Hierbij zijn de bevindingen uit tabel 3 gewogen. Bij het maken van de keuze is de programmeeromgeving belangrijker gevonden dan de leercurve die je door kunt maken bij het werken in de simulatietool.

RinSim voldoet aan heel veel van de randvoorwaarden en is geschikt voor deze opdracht. RinSim ondersteunt bijvoorbeeld (de)centrale algoritmen in combinatie met logistieke problemen, gegevensverzameling en het opslaan en laden van scenario's. Ook heeft bij het creëren van de simulator de focus op eenvoud en consistentie gelegen, wat in de leercurve terug te zien is. Wat de doorslag heeft gegeven is dat deze tool is ontwikkeld op universitair niveau en er bij de ontwikkeling centraal heeft gestaan om wetenschappelijk correcte gegevens te genereren. Verder heeft het een voordeel dat het is geschreven in Java, het volledig open-source en zeer goed gedocumenteerd is.

Het is dus best mogelijk dat sommige onderdelen buiten RinSim worden gemaakt, zoals bijvoorbeeld de software die is bedoeld voor het genereren van testbestanden.

[1] Programmeertalen te gebruiken via extension

## 5.3 Beperkingen van de grid lay-out

De volgende sprint had als onderwerp het onderzoeken van eventuele beperkingen van de grid lay-out die zijn gebruikt in de paper (van Moergestel et al., 2018). Een belangrijk deel van de opdracht is het verbeteren van de efficiëntie van het agent controlled productiegrid. Om dit te kunnen verbeteren moet eerst onderzocht worden wat de beperkingen zijn. In deze sectie zal daar antwoord op gegeven worden.

Om producten te produceren moet een transportplatform de start- en tussenproducten verplaatsen tussen de Equiplets. Als de Equiplet bezig is de productiestappen uit te voeren moet het transportplatform daarop wachten totdat deze door kan naar de volgende Equiplet. Doordat wachtende transportplatforms de weg blokkeren moeten de overige transportplatforms constant omleidingsroutes zoeken. In de paper (van Moergestel et al., 2018) is zelfs beschreven dat transportplatforms rondjes rond een bezette Equiplet aan het rijden waren op zoek naar een manier deze te bereiken. Een ander probleem is dat de bi-directionele paden -zoals gebruikt in alle voorgaande simulaties- geen tegemoetkomend verkeer aan konden. Dit wil zeggen dat het feitelijk een eenrichtingverkeer weg is als er een transportplatform overheen rijdt.

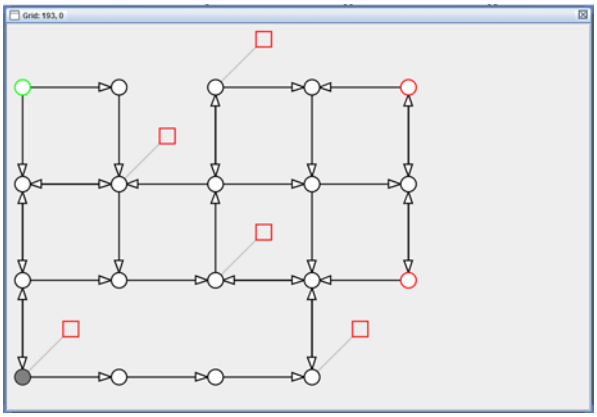
Er is ook nog onderzoek gedaan naar eventuele alternatieven voor de transportplatforms zoals de wat meer traditionele transportmethoden zoals transportbanden. Deze methoden hebben vaak het voordeel dat ze (veel) sneller zijn in het transporteren, echter zijn ze minder flexibel (Quinn et al., 1996; Leo van Moergestel et al., 2014). In overleg met de opdrachtgever is besloten om de traditionele transportmethode niet verder te onderzoeken maar me verder te richten op de simulatie.

Om oplossingen voor deze beperkingen te kunnen onderzoeken in de simulatie zal er een grid lay-out uitgezocht moeten worden, hier gaat de volgende sectie uitgebreid op in.

### 5.3.1 Keuze grid lay-out

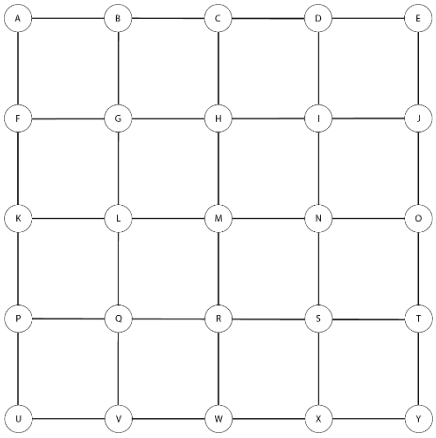
Een belangrijk onderdeel van de simulatie is het grid waarbinnen de transportplatforms zich moeten bewegen en waar de Equiplets werkzaam zijn. Omdat er in dit project alleen het paper (van Moergestel et al., 2018) met resultaten die zijn bevonden door Leo van Moergestel ter beschikking is gesteld en niet de oude simulaties zullen eerst de resultaten moeten worden gereproduceerd. Om deze resultaten te reproduceren moet een basis grid lay-out worden ontworpen. Voor deze basis lay-out is er gekozen voor een grid lay-out van een grid met op alle nodes een Equiplet, dit is besloten na overleg (van Moergestel & Wagensveld, 2020a). Deze is echter wel anders dan het grid waar Leo van Moergestel zijn resultaten mee heeft gegenereerd.

De door Leo van Moergestel gebruikte grid lay-out is getoond in figuur 2. De keuze om een andere lay-out te gebruiken dan gebruikt in de paper is gemaakt omdat er dan makkelijker doorontwikkeld kan worden naar volgende iteraties. Resultaten kunnen dan toch vergeleken worden met deze “nieuwe” basis lay-out. In figuur 3 is te zien hoe deze lay-out er ongeveer uit zou moeten zien.

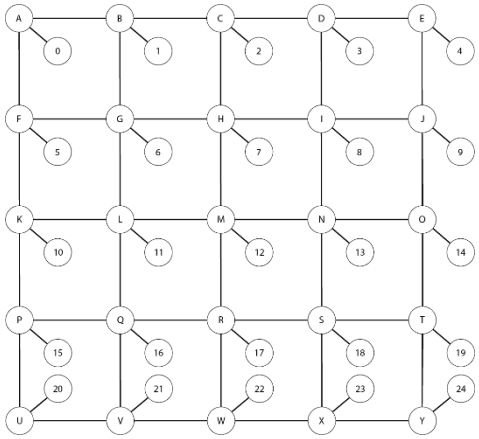


*Figuur 2: Example Grid lay-out*
*(van Moergestel, Puik, & Meyer, 2018)*

Als tweede lay-out van het grid is in overleg met Leo van Moergestel besloten om een grid te maken waarbij de transport agents “van het grid af” moeten om naar een Equiplet te gaan. Deze lay-out is te zien in figuur 4. Hiermee wordt voorkomen dat transport agents die een node moeten passeren zouden moeten wachten omdat deze node bezet is door een Equiplet die “bezig” is. Eventueel is in dit ontwerp ook nog buffer voor transport agents te realiseren. Ook hier zijn weer twee mogelijkheden om dit te realiseren, namelijk in de vorm van een stack of een queue.




*Figuur 3: "nieuwe" basis lay-out*




*Figuur 4: Grid met "losse" equiplets*

Beperkingen van de grid lay-out zijn vooral dat als een transportplatform niet direct terecht kan bij een Equiplet hij de transportbewegingen van de andere transportplatformen gehinderd worden. Dit heeft voornamelijk als oorzaak dat er in het grid geen plek is gedefinieerd als “parkeerplaats” waar een transportplatform tijdelijk kan wachten zonder dat dit invloed heeft op verdere verplaatsingen in dit grid door andere transportplatforms.

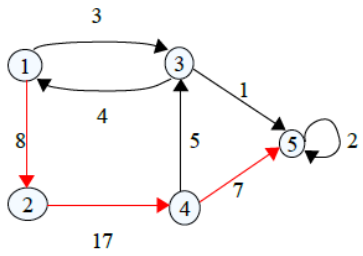
Een deel van de beperkingen is dat een transportplatform bij het binnengaan van het grid een lijst heeft met de Equiplets die hij moet bezoeken en niet met productiestappen die doorlopen moet worden. Doordat deze lijst vooraf is gedefinieerd liggen de stappen vast. Hierdoor is het systeem minder dynamisch, het zou beter zijn als de lijst productiestappen zou bevatten en dat op het moment dat een stap afgerond is er een (of meer) Equiplet(s) wordt gezocht bij de desbetreffende volgende stap en daarbij een route naar die Equiplet. Als er meerdere Equiplets zijn die de desbetreffende volgende productiestap uit kunnen voeren kan er zelfs gekeken worden welke van deze Equiplets het dichtste bij is.

Belangrijkste beperkingen van de grid lay-out zijn de invloed van “wachtende” transportplatformen door het ontbreken van parkeerplekken en de manier waarop het grid wordt gebruikt.

## 5.4 Beperkingen grid lay-out overkomen

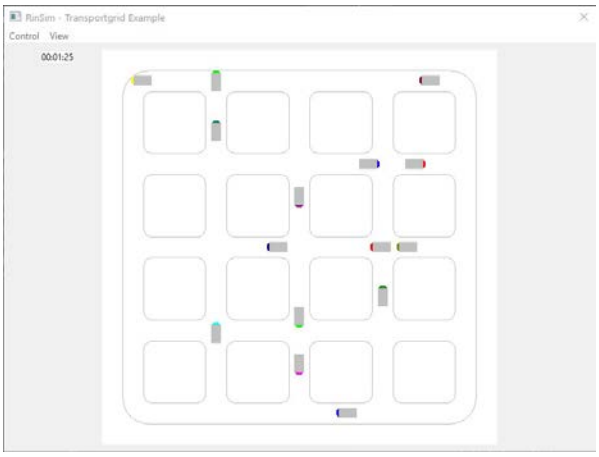
In deze sprint stond de vraag centraal hoe de invloed van de transportplatformen op het grid, zoals besproken in paragraaf 5.3.1, kan worden overkomen. Om deze vraag te kunnen beantwoorden is gebruik gemaakt van RinSim.

Het grid wordt beschreven met behulp van de grafentheorie (graphtheory), welke bekend is uit de wiskunde. Een graaf is een verzameling knopen (vertices of nodes in het Engels) die verbonden zijn met lijnen of zijden (edges in het Engels). Grafen kunnen gericht zijn (directed graph), dit betekent dat een zijde een specifieke richting heeft, dit kan gezien worden als “éénrichtingsweg”. Ook kunnen ze gewogen zijn, dat betekent dat een zijde een gewicht heeft. Door de gewichten van de zijdes op te tellen kan berekend worden welk pad door een graaf het zwaarst/duurst is. Zo is in figuur 5 (Kaldeway, 2016) een voorbeeld te zien van in pad in een graaf. Het pad loopt langs de volgende knopen 1 → 2 → 4 → 5. In dit geval is de ongewogen padlengte 3, namelijk het totaal aantal zijden. Echter de gewogen padlengte is 8+17+7 = 32.

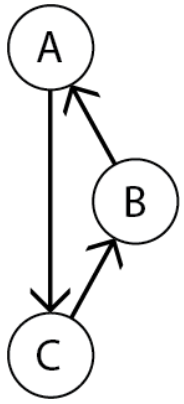



*Figuur 5: Voorbeeld graaf*

Om RinSim en de structuur te begrijpen volgt hier een korte uitleg. Binnen RinSim is een RoadModel een model dat een vloot aan voertuigen beheert in een (nog) ongedefinieerde omgeving. Vervolgens wordt in een GraphRoadModel een RoadModel uitgebreid met een graaf (graph) als wegennetwerk waarop voertuigen geplaatst kunnen worden. Het is ook mogelijk tijdens de simulatie zijden en knopen aan te passen, te verwijderen en te plaatsen.




*Figuur 6: Transportgrid 5X5*

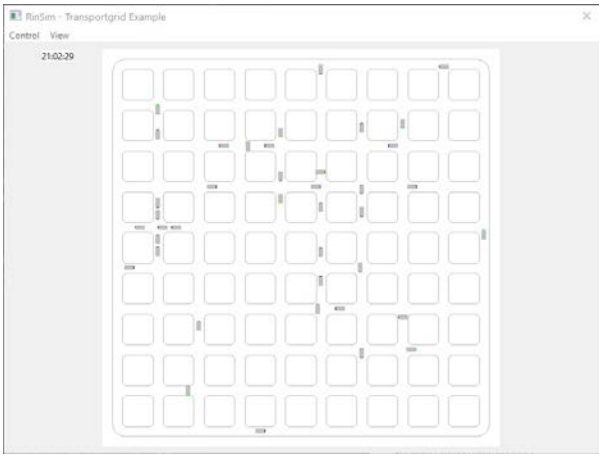



*Figuur 7:Voorbeeld*

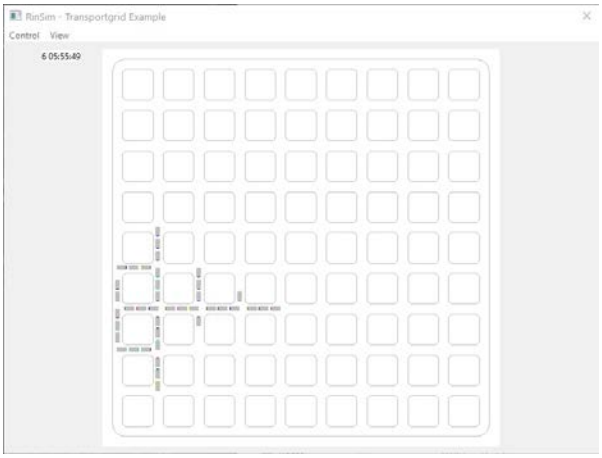
Met behulp van RinSim is eerst een grid van vijf bij vijf gemaakt. Hierin kunnen de transportplatforms zich heen en weer begeven, dit is te zien in figuur 6. Binnen RinSim is de meest voor de hand liggende optie het gebruik van het CollisionGraphRoadModel dit is een “GraphRoadModel” met botspreventie (R. van Lon, 2018a). Dit heeft namelijk de meeste functionaliteit die nodig is. Om dit grid te testen op gridlocks (en mogelijke andere eigenaardigheden) heb ik de transportplatforms willekeurig door het grid laten rijden. Echter bleek al snel dat als twee transportplatformen elkaar tegenkomen op een zijde zij stoppen om een botsing (collision) te voorkomen. Botsingen zijn te voorkomen door twee gerichte zijden te gebruiken, echter aangezien een volledig gerichte graaf (eenrichting grid) geen realistische optie is omdat paden en bereikbaarheid van de Equiplets daardoor onrealistisch lang worden. In RinSim blijkt in het “GraphRoadModel” geen mogelijkheid te zijn om twee gerichte zijden te maken in tegenovergestelde richting tussen twee knopen. In dat geval worden deze twee zijden samengevoegd tot één ongerichte zijde. Dit is op te lossen door één van deze twee zijden op te splitsen in twee gerichte zijden die half zo zwaar zijn (zoals weergegeven in figuur 7). Dit blijkt echter een theoretische oplossing aangezien de implementatie van de GraphRoadModel in RinSim zich strak aan een rechtlijnig grid houdt. Hierdoor komt de tussenliggende knoop (knoop B) weer op de zijde [A C] te liggen. Hierdoor detecteert een tranportplatform nog steeds een botsing bij een tegemoetkomend transportplatform op de andere zijde.

Om dit probleem te voorkomen moet dus de functionaliteit die zorgt voor de botspreventie worden aangepast. Echter is niet te achterhalen waar in de simulatie deze precies gedefinieerd is. Inmiddels op dit punt aangekomen blijkt de logische object georiënteerde opbouw van RinSim ook zijn achilleshiel te zijn. RinSim is namelijk opgebouwd uit een zeer groot aantal classes en interfaces die allemaal van elkaar overerven. Het blijkt dat de meer rudimentaire classes slecht gedocumenteerd zijn, dit is een probleem gebleken binnen dit afstudeeronderzoek.

Ook blijkt door de relatief kleine gebruikerscommunity dat het zoeken naar advies op het internet weinig uithaalt. Om dit te voorkomen adviseer ik ook bij het zoeken naar een andere simulatietool mede te zoeken naar een tool die op grotere schaal en door een grotere community gebruikt wordt.



*Figuur 8: Proefgrid 10X10*



*Figuur 9: Vastgelopen Proefgrid 10X10*

Om te kijken hoe groot het probleem van de gridlocks precies is heb ik ook een grid gemaakt van tien bij tien nodes, waarbij alleen de buitenste zijden ongericht zijn en de binnenste gericht. In figuur 8 is te zien hoe deze simulatie er uit ziet. Hieruit blijkt echter dat zelfs met een belasting van 5% de eigenschap om stil te gaan staan in plaats van een andere route te zoeken om een botsing te voorkomen het grid al vastloopt zoals te zien in figuur 9.

Nadat ik al deze informatie heb gewogen kom ik tot de conclusie dat het strak volgen van de lijnen van een grid niet het meest efficiënt is en voor complexe problemen kan zorgen. Ik zal hier verder op in gaan bij de aanbevelingen.

### 5.4.1 Keuze RinSim voor simulatie

In de laatste fase van mijn afstudeerproject ben ik tot de conclusie gekomen dat RinSim toch niet de ideale simulatietool is voor deze opdracht.

RinSim is niet de simulatietool gebleken die zonder enige kennis gebruikt kan worden. De documentatie lijkt in eerste instantie goed tot zeer goed te zijn. Echter wat vooraf niet duidelijk is, is dat om in RinSim een simulatie te maken het een aanzienlijke hoeveelheid werk vereist om een scenario zelf te maken. Omdat RinSim alleen algemene abstracte classes bevat, waardoor elk simulatiescenario goed op maat gemaakt kan worden voor zijn doel. Door deze segmentatie is RinSim “eenvoudig” uit te breiden en zeer krachtig te maken. Wat dit als zeer groot nadeel heeft is dat de je als ontwikkelaar van een simulatie feitelijk de volledige code-base van RinSim moet kennen. Tot mijn verbazing ontbreekt een handleiding die helpt bij het opzetten van simulatie in RinSim. De voorbeelden (examples) voorzien ook niet in voldoende uitleg. Hierdoor blijkt dus dat RinSim als tool niet geschikt is om zonder voorkennis aan rapid prototyping te doen. Ik ben toen vervolgonderzoek gestart om te kijken of er toch een alternatief was voor RinSim. Dit bleek MATSim te zijn. Als ik dit eerder had ontdekt, had ik de simulatie in MATSim uitgevoerd. Met deze kennis is de leercurve dan ook aangepast in de herziene beslismatrix zoals te zien in tabel 3.

### 5.4.2 MATSim

MATSim (Horni, Nagel, & W Axhausen, 2016a) is een in Java geschreven gratis en open-source multi-agent simulatie framework. MATSim is ontworpen door Kai Nagel en Kay W. Axhausen (Horni, Nagel, & W Axhausen, 2016b). De basis van de simulator komen van het tranportsimulatie platform TRANSIMS (Smith, Beckman, & Baggerly, 1995) en er zitten natuurkundige en civieltechnische aspecten in vanuit de achtergrond van Kay W. Axhausen (Horni et al., 2016b). Kenmerken van MATSim zijn dat het modulair is, het zeer grote scenario's kan simuleren, het zeer snel grote simulaties kan draaien en het is gericht op macroscopische scenario's. Op de website wordt zelfs gesteld dat de code-base onder constante ontwikkeling is. Na het bezoeken van de github repository (GitHub, 2020) lijkt dat wel te kloppen aangezien er het volgende te lezen was: "Excluding merges, 9 authors have pushed 8 commits to master and 24 commits to all branches. On master, 21 files have changed and there have been 576 additions and 402 deletions".

Ik heb vervolgens MATSim toegevoegd aan de beslismatrix, zie tabel 4.

Ondanks dat ik wel een conclusie heb getrokken heb ik deze niet kunnen onderbouwen met behulp van een simulatie. Uiteindelijk is het dus niet gelukt deze deelvraag te beantwoorden. Ik zal hier bij de aanbevelingen (hoofdstuk 7) verder op in gaan.

*Tabel 4: Herziene beslismatrix*

| | Programmeertaal | | | | Platform | | | | | | Algoritmen | | | Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Naam | C/C++ | Java | Python | Overige | Windows | Linux | macOS | Leercurve (0-7) | Open-Source | Betaald | Centraal | Decentraal | Keuze uit meerdere | Verkeer | Systeem | Logistiek | Sociale problemen |
| RePast Simphony | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 2-6 | ✓ | | | ✓ | ✓ | | | | ✓ |
| Simulink | | | | ✓ | ✓ | ✓ | | 6 | | ✓ | ✓ | | ✓ | | ✓ | | |
| Analytic Solver Simulation | | | | ✓ | ✓ | ✓ | ✓ | 5 | | ✓ | | | | ✓ | ✓ | ✓ | |
| RinSim | | ✓ | | | ✓ | ✓ | ✓ | 6[2] | ✓ | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| NetLogo | | ✓[3] | ✓[3] | ✓ | ✓ | | ✓ | 2-3 | ✓ | | | ✓ | ✓ | ✓ | ✓ | | ✓ |
| MESA | | | ✓ | | ✓ | ✓ | ✓ | 4 | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ |
| MATSim | ✓ | ✓ | | | ✓ | ✓ | ✓ | 3 | ✓ | | | ✓ | ✓ | ✓ | | | ✓ |

[2] Aangepast van 3 naar 6 met de huidige kennis.
[3] Programmeertalen te gebruiken via extension

## 5.5 Centrale padplanning agent

De vraag ‘Welke invloed heeft het gebruik van een centrale padplanning agent op deze beperkingen?’ heb ik niet kunnen beantwoorden met RinSim. Dit komt doordat ik niet de simulatie heb kunnen creëren zoals dat de bedoeling was. Om op deze vraag toch antwoord te kunnen geven is gekeken of er literatuur beschikbaar is die hier iets over zeggen. Rinde van Lon (R. R. van Lon & Holvoet, 2017) heeft zich dit blijkbaar ook afgevraagd en heeft hier ook onderzoek naar gedaan. Van Lon kwam tot de conclusie dat decentrale padplanning beter is dan centrale padplanning als er sprake is van situaties met zowel middel tot hoge dynamiek, hoge urgentie en middel tot hoge schaal. Het gecentraliseerde algoritme vond in deze gevallen een oplossing die 112,3% kostte ten opzichte van een gedecentraliseerd algoritme. Maar in andere gevallen is een gecentraliseerd algoritme wel beter, met een koste van 94,2%.

Dit maakt de conclusie (van Moergestel et al., 2018) die getrokken is door Leo van Moergestel extra interessant om te bewijzen. Zeker om te zien of het probleem van de deadlock ligt bij de manier van padplanning of dat er juist een andere oorzaak is, zoals de architectuur van het productiegrid.

# 6 Proof of Concept

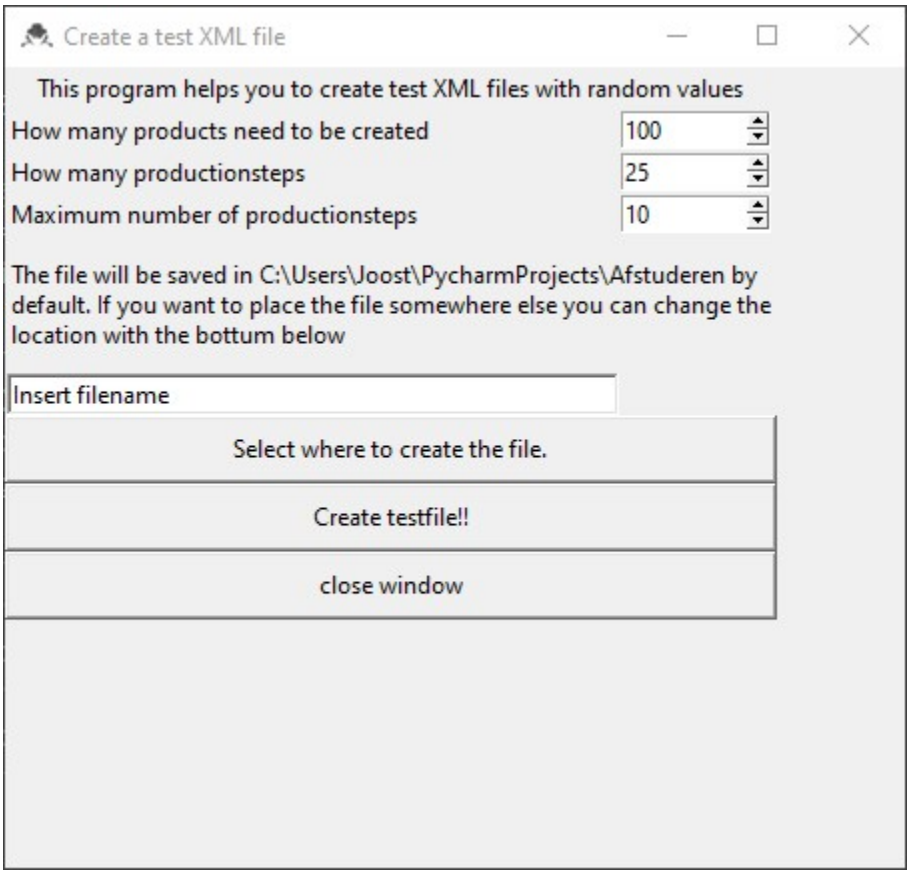
In dit hoofdstuk zal ingegaan worden op de opgeleverde onderdelen die samen het proof of concept zullen vormen en de onderbouwing zullen leveren voor de antwoorden op de deelvragen. Deze onderdelen zullen allemaal ook terug te vinden zijn via GitHub (Wagensveld, 2020).

## 6.1 XML testbestanden

De basis voor de simulatie is een lijst van producten die gemaakt moeten worden door het gesimuleerde agile manufacturing productiegrid. Om de simulatie goed te kunnen testen zouden er meerdere lijsten gebruikt moeten worden. Leo van Moergestel heeft een voorbeeld van een dergelijke lijst gedeeld. Deze lijst was opgesteld in het XML-format. Om andere lijsten te maken heb ik in Python het volgende script geschreven. Dit is onderdeel van een groter geheel waardoor er in een GUI-Window via een aantal invoervelden een XML-bestand gecreëerd wordt met daarin random waarden.

```
def createFile():
    global NUM_OF_LINES
    global MAX_NUM_OF_STEPS
    global STEP_RANGE
    global FILENAME
    global SAVEPATH

    NUM_OF_LINES = int(linesbox.get())
    MAX_NUM_OF_STEPS = int(stepsbox.get())
    STEP_RANGE = int(rangebox.get())

    FILENAME = os.path.splitext(filenameField.get())[0]
    FILENAME += ".xml"

    if os.path.exists(FILENAME):
        os.remove(FILENAME)  # this deletes the file
    else:
        print("The file does not exist (yet), so nothing deleted")  # added this to prevent errors

    print("SAVEPATH = " + SAVEPATH + " and FILENAME = " + FILENAME)

    f = open(SAVEPATH + FILENAME, "a+")
    f.write("<?xml version='1.0' encoding='utf-8'?>\n"
            "<root>\n")
    TEMPLATE = '<P>P{}<NPS>{}</NPS><STEPS>{}</STEPS></P>'
    for i in range(0, NUM_OF_LINES):
        steps = random.randint(1, MAX_NUM_OF_STEPS)
        step_values = [str(random.randint(0, STEP_RANGE)) for x in range(0, steps)]
        line = TEMPLATE.format(i, steps, ','.join(step_values))
        f.write("\t" + line + "\n")
    f.write("</root>")
    f.close()
    messagebox.showinfo("File Created", "\"" + FILENAME + "\" saved in \"" + SAVEPATH + "\"")
```

*Figuur 10: Functionaliteit om XML-testbestanden te maken*




*Figuur 11: Simulatietool*

De code zoals weergegeven in figuur 10 is de functionaliteit die achter het programma in figuur 11 zit. Als er gekozen wordt voor 'Create testfile!!' wordt het volgende XML-bestand gegenereerd, deze is te zien op de volgende bladzijde (figuur 12).

```
<?xml version='1.0' encoding='utf-8'?>
<root>
    <P>P0
        <NPS>8</NPS>
        <STEPS>5,9,10,5,2,4,7,9</STEPS>
    </P>
    <P>P1
        <NPS>18</NPS>
        <STEPS>4,7,9,3,1,0,2,10,4,3,8,2,9,4,0,2,9,5</STEPS>
    </P>

[…]

    <P>PX
        <NPS>4</NPS>
        <STEPS>7,7,4,1</STEPS>
    </P>
</root>
```

*Figuur 12: Opbouw XML testbestand*

Figuur 12 laat het XML-bestand zien. Dit XML-bestand is als volgt opgebouwd. Binnen de <root> tags is een lijst gedefinieerd met te maken producten. Binnen de <P> tags is dit product gedefinieerd waar de tag eerst gevolgd wordt door de unieke productID. Dit wordt gevolgd door het aantal productiestappen dat nodig is om het product te vervaardigen, dit staat gedefinieerd tussen de <NPS> tags. Vervolgens staan tussen de <STEPS> tags deze stappen gedefinieerd. Tijdens de simulatie zijn de nummers van de productiestappen gelijk aan het ID van Equiplets. In werkelijkheid zal bij een productiestap gekeken worden welke productiestap gedaan kan worden door welke Equiplet en kunnen twee verschillende achtereenvolgende productiestappen mogelijk ook door één Equiplet uitgevoerd worden zonder dat het product daarvoor verplaatst hoeft te worden. De product definitie wordt afgesloten door de </P> sluittag, waarna het volgende product op dezelfde manier staat gedefinieerd.

# 7 Conclusie en aanbevelingen

Dit hoofdstuk beschrijft de conclusies en aanbevelingen die opgesteld zijn naar aanleiding van dit afstudeeronderzoek.

Bij deze afstudeeropdracht is onderzocht welke oplossingen er zijn om door middel van simulaties de efficiëntie van een Agile manufacturing productiegrid de productie te verbeteren.

Allereerst is er bepaald aan welke randvoorwaarden de simulatietool moet voldoen. Hierbij is gekeken naar de programmeertaal, het platform, de gebruikte algoritmen en het type simulatietool.

Hieruit bleek dat RinSim de simulatietool leek die aan de randvoorwaarden voldeed en geschikt leek voor deze opdracht.

De belangrijkste beperkingen van de grid lay-out in de paper (van Moergestel et al., 2018), waren de invloed van “wachtende” transportplatformen door het ontbreken van parkeerplekken en de manier waarop het grid wordt gebruikt.

Deze beperkingen kunnen overkomen worden door niet strak de lijnen van het grid te volgen, zodat een object op dit grid geen invloed heeft op de overige bewegingen in dit grid.

Een centrale padplanning agent is efficiënter dan decentrale aansturing als er geen sprake is van situaties met zowel.

Door deze deelvragen te beantwoorden, kan de hoofdvraag ‘Welke oplossingen zijn er om door middel van simulaties de efficiëntie van een Agile manufacturing productiegrid de productie te verbeteren ter validatie van de theorie (van Moergestel et al., 2018)?’ beantwoord worden:

Centrale padplanning is inderdaad efficiënter, zoals de theorie (van Moergestel et al., 2018) stelt, zolang er niet wordt voldaan aan de volgende drie criteria:

- middel tot hoge dynamiek in het productiegrid
- hoge urgentie binnen het productiegrid
- middel tot hoge schaal binnen het productiegrid.

Ook is de manier waarop het huidige grid is vormgegeven niet efficiënt omdat obstakels te veel invloed hebben of transportbewegingen.

## Aanbevelingen

Mijn advies zou dan ook zijn om het transportgrid te gaan zien als open veld waarbinnen de transportplatformen zich vrij kunnen bewegen. Hierdoor is het uitwijken voor obstakels geen probleem, het maakt dan zelfs niet uit of een obstakel stationair is of bewegend. Qua voorrangsregels zou dan een voorbeeld genomen kunnen worden aan de scheepvaart. Een open veld productie omgeving heeft mogelijk ook als voordeel dat rondom Equiplets waarvan bekend is dat deze drukker bezocht worden er ruimte vrijgehouden kan worden tot andere Equiplets.

# 8 Evaluatie

Ik ben erg tevreden over hoeveel ik heb geleerd over verschillende padplanning algoritmen (ook al heb ik er maar één gebruikt in mijn simulatie), hoe deze ingezet kunnen worden binnen netwerken en ook binnen agent controlled production grids. Ook heb ik veel geleerd over fysieke aanpassingen die gedaan kunnen worden om logistieke problemen op te lossen. Achteraf gezien heb ik misschien iets te veel tijd gestoken in het zoeken naar (wetenschappelijke) literatuur dan in het aantonen van de gestelde conclusie in het artikel van Leo van Moergestel. Ik vind het jammer dat het niet gelukt is om de planning te volgen.

## 8.1 Kwaliteit

Ik heb geprobeerd voor zover mogelijk de Software Rules (van Ooijen, 2014) aan te houden. Dit is niet altijd mogelijk geweest omdat Wouter van Ooijen deze regels specifiek heeft opgesteld voor het schrijven van herbruikbare embedded C/C++ code. Waar deze set regels niet toepasbaar zijn geweest heb ik geprobeerd wel zo veel mogelijk het idee achter deze regels aan te houden. Verder heb ik alle gemaakte software in een Git repository geplaatst (Wagensveld, 2020) zodat deze makkelijk teruggevonden kunnen worden als er in de toekomst iemand een vervolgonderzoek wil doen.

Ik heb mijn scriptie als verslaglegging van de afstudeerstage zelf opgesteld aan de door de Hogeschool Utrecht opgestelde kwaliteitseisen zoals deze terug te lezen zijn in de afstudeerleidraad (Kaldeway, 2019).

## 8.2 Risico's

Achteraf gezien heb ik het risico van het te weinig bijhouden van mijn programmeervaardigheden onderschat. Ik heb gemerkt dat ik veel meer tijd heb moeten stoppen in het programmeren dan ik van tevoren had verwacht, ik moest er echt weer "in komen" om het heel simpel te zeggen.

Wat verder erg vervelend was dat ik de eerder gemaakte simulaties nooit heb gekregen en ik dus alles zelf heb moeten bedenken/programmeren. Ik hoopte de resultaten van eerdere simulaties, de omschrijvingen van eerdere gesimuleerde modellen en XML-beschrijvingen van het productieprocessen van 'voorbeeld'producten te krijgen. Het enige wat ik heb gekregen is een voorbeeld van een XML test bestand. Ik heb meerdere keren om deze simulaties gevraagd bij de opdrachtgever, achteraf gezien had ik dit moeten escaleren richting de docentbegeleider.Ik had ingeschat dat een risico zou zijn dat de simulatie niet stabiel zou zijn, dit heb ik weten te ondervangen door RinSim te gebruiken, hierdoor wist ik dat dit stabiel zou draaien. Ik heb me hierdoor alleen hoeven richten op de functionaliteit.

Bij aanvang van het afstuderen dacht ik dat de heupoperatie van mijn vrouw een vertragende factor zou zijn. Doordat ik veel langer heb gedaan over het afstuderen heeft de heupoperatie van mijn echtgenote geen (echte) invloed gehad. Wel gaf de operatie mij een excuus om even niet na te hoeven denken over de scriptie. Dit heeft uiteindelijk wel vertragend gewerkt helaas.

## 8.3 Persoonlijke ontwikkeling

Ik heb met heel veel plezier mijn kennis van JAVA en PYTHON weer bijgehaald met deze opdracht waardoor hij naar mijn idee erg goed past bij mijn studie Technische Informatica. De belangrijkste technische uitdaging vond ik om de oude simulaties die in een oudere opstelling zijn gedaan voor dit principe te reproduceren. Op gebied van de andere doelen die ik mezelf had gesteld ben ik erg tekortgeschoten. Deze zal ik nu met behulp van de STARR-methode drie ervaringen en situaties uitwerken.

| 1. Van toepassing op de PS Onderzoekend probleem oplossen en doelgericht interacteren |
|---|
| **Situatie** Bij mijn afstudeeronderzoek "Project Productiepadplanning 2.0 in een Agile Multiparallel Productiegrid" heb ik gedurende negen maanden gewerkt aan het vinden van een simulatietool voor een Agile Productiegrid. Ik werkte hier alleen aan, vanuit huis, en had weinig mogelijkheden om te overleggen met anderen. Ik werd op afstand begeleid door de opdrachtgever. Ik werd hierin aangesproken op de deelvaardigheiden Samenwerken en Problemen oplossen.<br>**Taak.** Ik moest door middel van literatuuronderzoek een simulatietool selecteren die ik kon gebruiken voor mijn afstudeeronderzoek.<br>**Actie.** Ik heb hier geprobeerd te overleggen met de opdrachtgever. Ook heb ik zelf geprobeerd het probleem op te lossen, door de literatuur te lezen en er zelf over na te denken, en verschillende simulatietools te onderzoeken. Ik kwam toen al redelijk snel tot de conclusie dat RinSim de beste simulatietool voor mij was.<br>**Resultaat.** Echter, aan het eind van het project bleek dat een andere simulatietool waarschijnlijk een betere keus was geweest. Ik heb hierdoor geen werkende simulatie toe kunnen voegen aan mijn scriptie.<br>**Reflectie.** Voor toekomstige projecten moet ik dus uitkijken dat ik niet te snel conclusies trek, en eerst alle opties naast elkaar zetten. Ook is het verstandig om deze opties nog meer met iemand anders te bespreken, om zo mijn bevindingen te spiegelen en te kijken of ik niet te snel een conclusie trek. Of dat er misschien nog andere aspecten zijn die ik mee zou moeten nemen in het trekken van een conclusie. |

| 2. Van toepassing op PS Toekomstgericht organiseren en Persoonlijk leiderschap |
|---|
| **Situatie.** Ik startte mijn afstudeerproject aan het begin van de coronacrisis. Het plan voor mijn afstuderen wat ik al anderhalf jaar had, en wat ook uitvoerig besproken was met mijn begeleiders, kon de prullenbak in. Doordat de Hogeschool gesloten was voor studenten kon ik de geplande opdracht niet uitvoeren.<br>**Taak.** Mijn taak was om een afstudeerproject uit te voeren, binnen de op dat moment geldende beperkingen. Eigenlijk was het ook mijn taak om binnen de gestelde tijd dit project uit te voeren. Ik weet dat ik op het deelgebied Werkzaamheden managen nog winst kan behalen. Binnen dit onderdeel was ook het deelgebied Ondernemend zijn van toepassing.<br>**Actie.** Ik heb een tijdsplanning gemaakt, waarbij ik rekening heb gehouden met barrières en persoonlijke omstandigheden. Door te plannen heb ik getracht de tijd te bewaken. Echter, doordat we eerst een nieuwe opdracht moesten bedenken, en ik hier enthousiast over moest worden, heeft dit al de eerste anderhalve maand vertraging opgelopen. Ik vond het lastig mijzelf te motiveren voor de nieuwe opdracht.<br>**Resultaat.** Het is me niet gelukt me aan mijn eigen planning te houden. Uiteindelijk heb ik drie maanden langer over mijn afstuderen gedaan. De beperkingen door de coronacrisis waren dusdanig dat ik sterk belemmerd werd in mijn werkzaamheden.<br>**Reflectie.** Ik vind het vervelend dat ik vertraging heb opgelopen. Voor toekomstige projecten wil ik vaker met de opdrachtgever afstemmen over de planning, en afspraken maken over de oplevering van deelopdrachten. |

3. Van toepassing op PS Doelgericht interacteren

**Situatie.** Wat ook tot vertraging heeft geleid is dat ik niet makkelijk heb kunnen “sparren” bij het onderzoeken/uitwerken van bepaalde ideeën en functionaliteiten. Ik miste contactmomenten met mijn begeleiders, en het eenvoudig iets kunnen vragen doordat je in hetzelfde gebouw bent. Ik merkte bij mijzelf dat het fysiek op afstand zijn voor een wat betreft het vragen om hulp. Ook vond ik het lastig dat er geen vast (wekelijks) overlegmoment was, en dat er beperkt tijd was voor de begeleiding.
**Taak.** Mijn taak als student was om de afstudeeropdracht uit te voeren. Hiervoor was goede communicatie met de opdrachtgever, procesbegeleider en docentbegeleider nodig.
**Actie.** Ik heb wel getracht aan te geven via de mail dat ik contact nodig had.
**Resultaat.** Toen een reactie uitbleef op een aantal door mij gestuurde e-mails, heb ik het erbij laten zitten. Wat ik wel heel erg fijn heb gevonden is de “informele” vorm van communicatie via Whatsapp dit ik met de opdrachtgever heb gehad.
**Reflectie.** Achteraf gezien had ik hier meer om moeten vragen, en had ik moeten aangeven dat het voor mij heel belangrijk is –ook om de voortgang te waarborgen- om wekelijks contact te hebben. Ook had ik bij mijn andere begeleiders aan de bel moeten trekken, of moeten proberen op een andere manier in contact te komen. Dit neem ik mee naar de toekomst, als ik bij een werkgever mogelijk ook op afstand moet werken.

## 8.4 Ethische afweging

In deze paragraaf zal ik een ethisch dilemma waar ik tijdens mijn afstudeerproject tegenaan liep beschrijven. Ook ga ik in op hoe ik in de toekomst met ethische afwegingen om wil gaan.

4. Ethisch dilemma

**Situatie.** Wij hebben als mensheid de verantwoordelijkheid op een correcte manier met de natuurlijke (hulp)bronnen van de wereld om te gaan. Dat is actueler dan ooit in een tijd waarin we te maken hebben met stikstof-, $CO_2$-, handels- en corona-crises. Ook is er, aan de andere kant, de maatschappelijke discussie rondom robotica en werkeloosheid rond deze ontwikkeling die een rol kan gaan spelen.
**Taak.** Mijn taak als student was om binnen de afstudeeropdracht te onderzoeken welke opties er zijn voor een agent controlled productiegrid.
**Actie.** Ik heb in de literatuur gezocht welke voor en nadelen er zijn wat betreft Agile produceren. Ik merkte dat ik het best lastig vond om hier objectief naar te kijken, en goed zowel voor- als nadelen te vinden.
**Resultaat.** Agile Manufacturing en zou ertoe kunnen leiden dat er veel meer gerecycled kan worden, dat er meer lokaal geproduceerd wordt en dat er veel minder grondstoffen verspild worden (van Moergestel, 2014). Agile Manufacturing zouden er daarom toe kunnen leiden dat het productieproces groener wordt, dit is een voordeel van deze manieren van produceren.
Er is een kans dat er door het toepassen van Agile Manufacturing en Intelligent Manufacturing banen verloren gaan doordat processen door robots/agents overgenomen kunnen worden. Dit is een nadeel van deze manier van produceren.
**Reflectie.** Ik heb gemerkt dat er zowel voor- als nadelen kunnen zijn bij overgaan op Agile Manufacturing. Dit is een dilemma, waar ik zelf geen antwoord op heb. Ik had hier graag met mijn begeleiders en andere studenten op school over gesproken, door de geldende maatregelen kon dat helaas niet.

Bij toekomstige werkgevers zal ik in gesprek gaan met de opdrachtgever om samen te bespreken welke ethische dilemma's er zijn, en hoe we hier mee om willen gaan. Ook zal ik zorgen dat ik met meerdere mensen hierover kan spreken, als ik het idee heb dat ik er zelf niet uitkom, of als ik moeite heb met de uitkomst.

## 8.5 Eindconclusie reflectie

Bij dit afstudeeronderzoek heb ik gemerkt dat ik een aantal zaken goed gingen, en ik een aantal struikelpunten had. Zo had ik moeite met het werken op afstand, de moeite die het kostte om contact te krijgen met mijn begeleiders, had ik problemen met mijn planning en trok ik te snel conclusies. Daarom heb ik de volgende leerdoelen opgesteld voor de toekomst:

- Persoonlijk leiderschap
    - Ik ben bij mijn toekomstige werkgever in staat op afstand te werken.
- Onderzoekend Probleem Oplossen
    - Ik ben in staat op basis van wetenschappelijke bronnen een gewogen beslissing te nemen.
- Doelgericht interacteren
    - Ik kan na drie maanden bij mijn toekomstige werkgever duidelijk communiceren naar mijn directe collega's.
- Toekomstgericht organiseren:
    - Ik kan aan het einde van mijn afstudeerperiode een reële planning maken van mijn werk waardoor ik de gestelde doelen haal.

# 9 Bronvermelding


Abelson, H., Goodman, N., & Rudolph, L. (1974). Logo manual.

Agile Methodologies for Industrial Production. (2020). Agile Methodologies for Industrial Production. Retrieved from https://www.theindustriallean.com/concepts/agile-methodologies-for-production/

Aldewereld, H. M. (2020, 8 april 2020). [Email conversation].

Andriessen, D. (Producer). (2014, 13-05-2020). Praktisch relevant én methodisch grondig? Dimensies van onderzoek in het HBO. [PDF] Retrieved from http://www.onderzoekscoach.nl/wp-content/uploads/2014/04/Openbare-Les-Daan-Andriessen1.pdf

Dean, J. S., Gumerman, G. J., Epstein, J. M., Axtell, R. L., Swedlund, A. C., Parker, M. T., & McCarroll, S. (2000). Understanding Anasazi culture change through agent-based modeling. *Dynamics in human and primate societies: Agent-based modeling of social and spatial processes*, 179-205.

Frontline Systems Inc. (2020). What's the Easiest Way to Solve Optimization Problems? | solver. Retrieved from https://www.solver.com/whats-easiest-way-solve-optimization-problems

GitHub. (2020). Pulse · matsim-org/matsim-libs. Retrieved from https://github.com/matsim-org/matsim-libs/pulse

Google Scholar. (2020). Google Scholar. Retrieved from https://scholar.google.nl/

Hogeschool Utrecht. (2020a). De HU is een inspirerende kennis- en innovatiepartner | Hogeschool Utrecht. Retrieved from https://www.hu.nl/onderzoek

Hogeschool Utrecht. (2020b). Lectoraat Microsysteemtechnologie | Hogeschool Utrecht. Retrieved from https://www.hu.nl/onderzoek/microsysteemtechnologie

Horni, A., Nagel, K., & W Axhausen, K. (2016a). *The multi-agent transport simulation MATSim*. London: Ubiquity Press.

Horni, A., Nagel, K., & W Axhausen, K. (2016b). The multi-agent transport simulation MATSim. In (pp. 3-8). London: Ubiquity Press.

Intelligent Systems Group. (2015). Intelligent Systems. Retrieved from http://www.cs.uu.nl/groups/IS/

Jennings, N. R., & Bussmann, S. (2003). Agent-based control systems. *IEEE control systems, 23*(3), 61-74.

JetBrains s.r.o. (2020). IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. Retrieved from https://www.jetbrains.com/idea/

Kaldeway, J. (2016). Reader Algoritmen en Datastructuren. In (Vol. 2020, pp. 175). Utrecht: Institute for ICT, HU.

Kaldeway, J. (2019). AFSTUDEERLEIDRAAD BACHELOROPLEIDING HBO-ICT VOLTIJD / DEELTIJD / DUAAL STUDIEJAAR 2019 - 2020. In (pp. 49): Hogeschool Utrecht.

Kazil, J., Pike, T., Masad, D., Garson, J., Mitchell, D. J., Hoover, A., . . . Ells, K. (2020, Jul 24, 2020). Mesa: Agent-based modeling in Python 3+. Retrieved from https://mesa.readthedocs.io/en/master/index.html

Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A top down approach* (Seventh edition. ed.). Hoboken, New Jersey: Pearson.

North, M. J., Collier, N. T., Ozik, J., Tatara, E. R., Macal, C. M., Bragen, M., & Sydelko, P. (2013). Complex adaptive systems modeling with Repast Simphony. *Complex adaptive systems modeling, 1*(1), 3.

Ozik, J., Collier, N. T., Murphy, J. T., & North, M. J. (2013). *The ReLogo agent-based modeling language.* Paper presented at the 2013 Winter Simulations Conference (WSC).

Puik, E., & van Moergestel, L. (2010). *Agile multi-parallel micro manufacturing using a grid of equiplets.* Paper presented at the International Precision Assembly Seminar.

Quinn, R. D., Causey, G. C., Merat, F. L., Sargent, D. M., Barendt, N. A., Newman, W. S., . . . Sterling, L. S. (1996). *Design of an agile manufacturing workcell for light mechanical applications.* Paper presented at the Proceedings of IEEE international conference on robotics and automation.

Railsback, S. F., & Grimm, V. (2019). *Agent-based and individual-based modeling: a practical introduction*: Princeton university press.
Raychaudhuri, S. (2008). *Introduction to monte carlo simulation.* Paper presented at the 2008 Winter simulation conference.
Redwood Logistics. (2020). Difference Between Lean and Agile Manufacturing – Redwood Logistics : Redwood Logistics. Retrieved from https://www.redwoodlogistics.com/difference-between-lean-and-agile-manufacturing/
Smith, L., Beckman, R., & Baggerly, K. (1995). *TRANSIMS: Transportation analysis and simulation system*. Retrieved from
Sniedovich, M. (2006). Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and cybernetics, 35*(3), 599-620.
Telgen, D. (2017). *Grid Manufacturing.* (PHD), Utrecht University, Utrecht.
Telgen, D., Puik, E., van Moergestel, L., Bakker, T., & Meyer, J.-J. (2015). Reconfigurable Equiplets Operating System A Hybrid Architecture to Combine Flexibility and Performance for Manufacturing. *International Journal on Advances in Software, 8*(3 & 4), 309-326.
The MathWorks Inc. (2020a). MATLAB - MathWorks - MATLAB & Simulink. Retrieved from https://nl.mathworks.com/products/matlab.html
The MathWorks Inc. (2020b). Simulink - Simulation and Model-Based Design - MATLAB & Simulink. Retrieved from https://nl.mathworks.com/products/simulink.html
van Lon, R. (2018a, Jun 11, 2018). AGV example | RinSim. Retrieved from https://rinsim.rinde.nl/learn/examples/agv/
van Lon, R. (2018b, 11-06-2020). RinSim: RinSim. Retrieved from https://rinsim.rinde.nl/
van Lon, R., & Holvoet, T. (2012). *RinSim: A simulator for collective adaptive systems in transportation and logistics.* Paper presented at the 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems.
van Lon, R. R., & Holvoet, T. (2017). When do agents outperform centralized algorithms? *Autonomous Agents and Multi-Agent Systems, 31*(6), 1578-1609.
van Moergestel, L. (2014). *Agent Technology in Agile Multiparallel Manufacturing and Product Support.* Utrecht University, Retrieved from https://dspace.library.uu.nl/handle/1874/298812
van Moergestel, L., Puik, E., & Meyer, J.-J. (2018, 24 - 28, 2018). *A Software Architecture for Transport in a Production Grid.* Paper presented at the INTELLI 2018, Venice, Italy
van Moergestel, L., Puik, E., Telgen, D., Kuijl, M., Alblas, B., Koelewijn, J., & Meyer, J.-J. C. (2014). *A simulation model for transport in a grid-based manufacturing system.* Paper presented at the Proc. of the Third International Conference on Intelligent Systems and Applications (INTELLI 2014).
van Moergestel, L., Telgen, D., Puik, E., & Meyer, J.-J. (2014). *Agent-based Manufacturing in a Production Grid*.
van Moergestel, L., & Wagensveld, J. P. (2020a, April 15 2020). [Telefonisch overleg 15-04-2020].
van Moergestel, L., & Wagensveld, J. P. (2020b, 26-05-2020). [Telefonisch overleg 26-05-2020].
van Moergestel, L., & Wagensveld, J. P. (2020c, 02-04-2020). [Telefonisch overleg over projectvoorstel 02-04-2020].
van Ooijen, W. (2014). *TI-C++-software-rules*. Retrieved from http://wagensveld.eu/TISoftwareRules.docx
Wagensveld, J. P. (2020). GitHub | Josque/Afstudeerproducten. Retrieved from https://github.com/Josque/Afstudeerproducten.git
Wilensky, U. (2016). NetLogo Home Page. Retrieved from https://ccl.northwestern.edu/netlogo/
Wilensky, U. (2019). NetLogo 6.1.1 User Manual: Extensions Guide. Retrieved from http://ccl.northwestern.edu/netlogo/faq.html

Wooldridge, M. J., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review, 10*(2), 115-152.

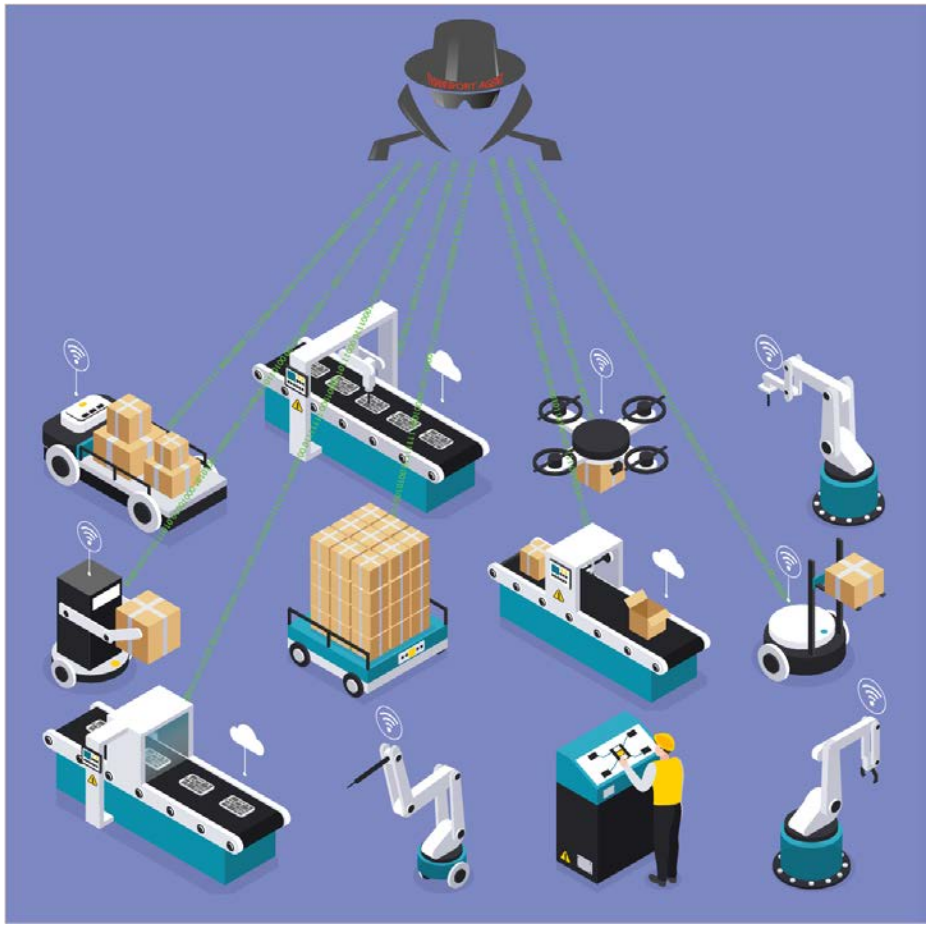
# Bijlagen

## Inhoud

# I. Functioneel Design (CAFCR)

# Project Productiepadplanning 2.0 in een Agile Multiparallel Productiegrid

*Functioneel document*



Joost Wagensveld-van Veen
1664713
Stagebegeleider HU:
Huib Aldewereld
Stagebegeleiders (bedrijf) HU:
Leo van Moergestel
Nini Salet
Versie 1.0

# MANAGEMENT SAMENVATTING

**Aanleiding**

Om in mijn afstudeerstage een goed eindproduct op te leveren moet een goed en gedegen functioneel ontwerp gemaakt worden. Daarvoor wordt dit CAFCR document gebruikt.

In opdracht van Leo van Moergestel is een ontwerp gemaakt voor een systeemsimulatie die de transportbewegingen en de daarbij voorkomende problemen gaat simuleren.

**Aanbevolen wordt:**

- een bestaande simulatie te gebruiken;
- gebruik te maken van RinSim.

**Motivatie**

Er worden meerdere bestaande simulatietools met elkaar vergeleken en daarbij worden de voors en tegens tegen elkaar weggezet.

**Consequenties**

Het eindproduct heeft een extra afhankelijkheid gekregen in de vorm van een bestaande tool.

# Inhoud

# 1. INLEIDING

“Efficiëntie, maar produceer met mate”, dat is het motto van het produceren met een agile productiegrid. Producten worden op afroep geproduceerd en dus niet op voorraad. Echter is de grote vraag hoe het transport in zo’n productiegrid het meest efficiënt georganiseerd kan worden. Die vraag staat centraal binnen dit afstudeerproject. Om dit te beantwoorden wordt in dit document een beschrijving gegeven hoe de simulatie gemaakt zal worden.

Aan de basis van dit document staat de CAFCR methodiek (Muller, 2004), dit omdat deze methodiek is geleerd tijdens het vak System Architecture (TCTI-V2SYAR-13). Er is geprobeerd deze methodiek zo veel mogelijk te volgen om het systeem te beschrijven. In hoofdstuk twee wordt de Customer Objectives view uitgewerkt. Vervolgens wordt in hoofdstuk drie de Application view uitgewerkt. In hoofdstuk vier wordt de Functional view uitgewerkt. Daarna wordt in hoofdstuk vijf de Conceptual view uitgewerkt. Ten slotte wordt de CAFCR methodiek afgesloten met in hoofdstuk zes de Realisation view. Na hoofdstuk zes zijn de referenties naar de verschillende bronnen benoemd en ook kunt u hier de bijlagen vinden.

# 2. CUSTOMER OBJECTIVES VIEW

Uit de Customer Objectives View moet duidelijk worden wie onze klant is en wat hij wil. Om daar achter te komen worden de stakeholders bepaald en geclassificeerd volgens het Mendelow's power-interest grid. Vervolgens worden de key-drivers bepaald.

## 2.1 Wie is de klant?

Om achter de stakeholders van dit project te komen wordt er gekeken naar vier groepen: users, developers, legislators en decisionmakers. Vervolgens wordt er per groep bekeken wie in dit project die groep representeert. Van elke groep worden vervolgens de stakeholders bepaald door te kijken naar suppliers, customers en satellite stakeholders.

De users van de simulatie zijn in dit geval Leo van Moergestel en Joost Wagensveld-van Veen, maar kunnen in de toekomst ook andere onderzoekers zijn die verder gaan in dit onderzoek. Zij zullen de simulatie gebruiken om de data in de paper (van Moergestel, Puik, & Meyer, 2018) te valideren.

Joost is de developer binnen dit project. De "klant" is Leo van Moergestel

De wetgever in Nederland is de 1e/2e kamer. Echter zullen die niet echt een rol spelen binnen dit project. Ook heeft de Hogeschool Utrecht, vertegenwoordigd door Joop Kaldeway, regels opgesteld in de vorm van de afstudeerleidraad. Deze regels worden gehandhaafd door de eerste en tweede examinatoren. Ook is Nini Salet betrokken bij het proces als zogeheten betrokken docent. Verder zijn er nog de familie, vrienden en het gezin van Joost die nauw betrokken zijn bij het behalen van het diploma.

De decisionmaker in dit project is Leo van Moergestel.

## 2.2 Welke stakeholders zijn het belangrijkst?

De stakeholders worden geclassificeerd op basis van twee belangrijke eigenschappen: macht en belang. Als een stakeholder veel macht heeft maar weinig belang dan moet ervoor worden gezorgd dat er voldaan wordt aan hun eisen en moet er geprobeerd worden meer interesse te kweken en hun richting de rechter kant te bewegen. Als een stakeholder veel belang heeft maar weinig macht dan kan er gebruik gemaakt worden van hun ideeën en interesse en als supporter van het project.

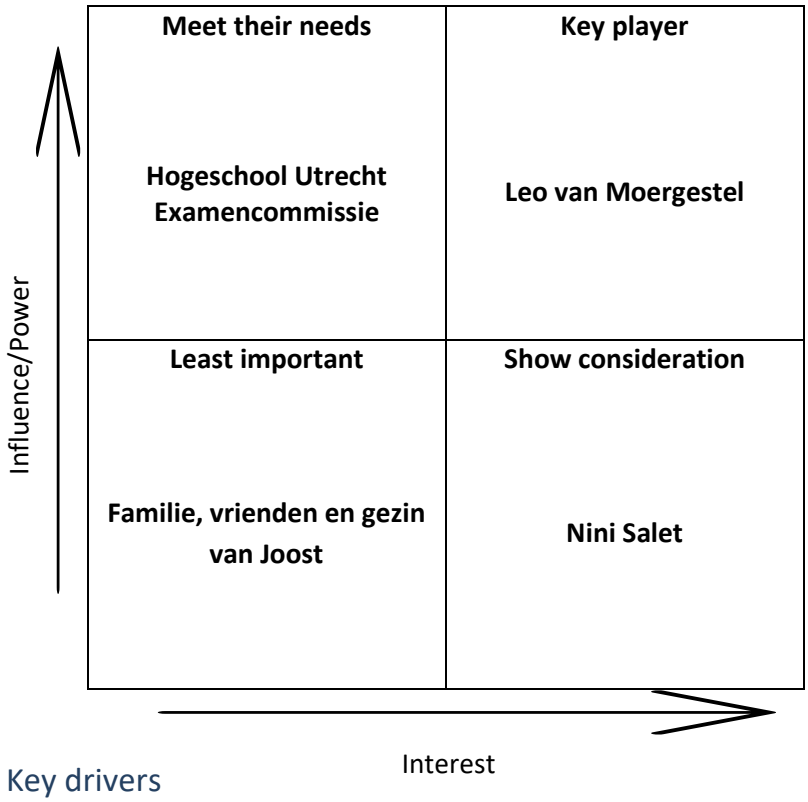
Heeft een stakeholder weinig belang en weinig macht dan zijn ze niet heel belangrijk maar moeten ze wel geïnformeerd blijven over het project om ze mogelijk meer interesse te kweken. Als een stakeholder veel macht heeft en veel belang dan is het een key player. Deze stakeholders hebben de focus in het project en moeten vaak worden geraadpleegd en betrokken worden bij de ontwikkeling.

In dit project zijn de vrienden van Joost niet heel belangrijk, zij hebben weinig macht en ook weinig interesse in de ontwikkeling van het project.

De 1e/2e kamer hebben veel macht maar weinig interesse, aan hun wetten zal moeten worden voldaan om geen problemen te krijgen. Ook de Hogeschool heeft in het project weinig interesse, maar zullen aan het eind wel het resultaat toetsen aan de door hun gestelde eisen.

Familie en het gezin van Joost en Nini Salet hebben veel interesse in het project, maar hebben weinig macht, zij zullen dus wel op de hoogte gehouden moeten worden.

De belangrijkste stakeholder is Leo van Moergestel, als opdrachtgever zal hij de meeste betrokkenheid en macht hebben. Er moet vaak worden overlegd om te kijken of de simulatie nog steeds voldoet aan het idee en eisen. Dit wordt uitgewerkt met behulp van het Mendelow's Power-Interest Grid zoals te zien hieronder.




## 2.3 Key drivers

Om de key drivers te bepalen wordt er gekeken naar de doelen van de stakeholders. Het doel van de opdrachtgever, Leo, is om de veronderstelde bevindingen uit zijn paper (van Moergestel et al., 2018) te kunnen valideren en mogelijk extra onderzoek te doen. Het doel van familie, vrienden en het gezin van Joost is om te zorgen dat Joost zijn opleiding afrondt. De Hogeschool Utrecht wil dat het diploma dat zij uitreiken aan het eind van de door Joost gevolgde opleiding een bepaalde waarde heeft. Dit leidt tot de volgende key drivers:

| Stakeholders | Doelen |
|---|---|
| Opdrachtgever (Leo) | Resultaten uit paper valideren |
| | Verder onderzoek mogelijk maken |
| | Uitzoeken wat de beste grid architectuur is |
| Familie, vrienden en gezin | Snel afgestudeerd zijn |
| Hogeschool Utrecht (examinatoren) | Een gedegen diploma kunnen uitreiken |

# 3 APPLICATION VIEW

## 3.1 Omgeving

De simulatie moet platform onafhankelijk kunnen worden uitgevoerd. Dit omdat er niet bekend is welk platform de toekomstige gebruikers zullen willen gebruiken. Er zijn vooralsnog geen constraints bekend die een bepaald platform afdwingen.

## 3.2 Application drivers

In dit hoofdstuk zal er verteld worden over de gevonden application drivers. Daarbij zijn de application drivers afgeleid van de key drivers van het vorige hoofdstuk.

### 3.2.1 Resultaten uit paper valideren

Het belangrijkste is dat de veronderstelde bevindingen uit het paper (van Moergestel et al., 2018) van bewijslast worden voorzien. Er moet dus een basis simulatie gedraaid worden en een simulatie met de in de paper voorgestelde aanpassingen.

### 3.2.2 Verder onderzoek mogelijk maken

Er moet aan de simulatie makkelijk dingen aangepast kunnen worden zodat andere situaties ook vergeleken kunnen worden.

### 3.2.3 Uitzoeken wat de beste grid architectuur is

Een belangrijke vraag die gesteld is in de opdracht is hoe het grid zo efficiënt mogelijk gemaakt kan worden. Bijvoorbeeld door richtingen van paden aan te passen of door inhaalstroken/uitvoegstroken te implementeren.

### 3.2.4 Gedegen diploma behalen

Als laatste doel van het project is een hoogwaardig/gedegen diploma behalen. Dit betekent dat alle opgedane kennis uit het gevolgde curriculum gebruikt zal worden om tot een hoogstaand eindresultaat te komen.

## 3.3 Customer story

**Meer soorten producten met minder verspilling**

*“Op het internet heb ik deze radio besteld. Ik kan precies aangeven welke kleur ik hem wil hebben en hij wordt helemaal individueel voor mij gemaakt, niet in China maar hier in Nederland. Ik zie precies wanneer de productie is gestart en wat de status is. Het mooiste is dat er voor mijn radio geen grote productielijnen hoeven te draaien. Er hoeft dus geen voorraad te worden bijgehouden, dit scheelt veel resources.*” Dit is een verhaal van iemand die online een bestelling op een productiegrid heeft geplaatst en omdat het allemaal automatisch gaat kan het goedkoop en lokaal geproduceerd worden. Maar hoe werkt dit nou

allemaal, hoe worden al die producten binnen het productiegrid efficiënt verplaatst. Daar is veel onderzoek aan vooraf gegaan.

# 4 FUNCTIONAL VIEW

## 4.1 Decompositie

In dit hoofdstuk zal worden beschreven hoe het systeem werkt en hoe het te bedienen is. Ook zal duidelijk worden uit welke onderdelen het systeem bestaat.

### 4.1.1 Systeem beschrijving

De simulatie wordt ontwikkeld om een werkende simulatie die op basis van een of meerdere XML-inputs de simulatie uitvoert en productiepaden genereert. Ook zal er een voorstel gedaan moeten worden hoe de grid architectuur eruit dient te zien om het grid zo efficiënt mogelijk te krijgen.

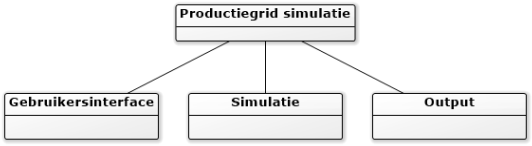
De resultaten zullen in een makkelijk te interpreteren bestand worden opgeslagen. Gegevens in dit bestand zullen bijvoorbeeld de tijdstippen en paden van het productieproces zijn. Deze output zal dan door middel van een script een grafiek of diagram genereren. Door middel van test sets met een voorspelbare uitkomst in de simulatie te draaien en te controleren of het systeem dat ook genereert kan de werking van het systeem gevalideerd worden (van Moergestel, 2020).

Doel van de opdracht is het verkrijgen van goede vergelijkingsresultaten om aan te tonen dat een overkoepelende transportagent de oplossing is voor capaciteitsproblemen in het transport binnen een productiegrid.

Het product dat opgeleverd zal worden is een simulatie, die voornamelijk resultaten zal tonen en niet een zeer geavanceerde grafische visualisatie.

### 4.1.2 Systeemonderdelen

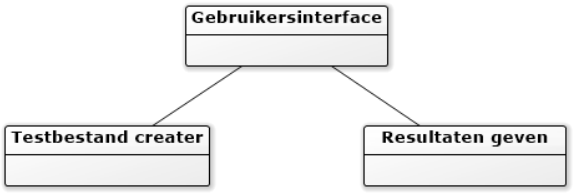
Om een systeem te kunnen maken zullen eerst de onderdelen moeten worden vastgesteld, dit kan gedaan worden met behulp van het bovenstaande hoofdstuk. Ten eerste wordt onderscheidt gemaakt in drie subsystemen, namelijk de simulatie, de gebruikersinterface en de output. Dit leidt tot de weergave die zichtbaar is in Figuur 1. Deze subsystemen zullen dieper worden uitgewerkt in de volgende hoofdstukken.



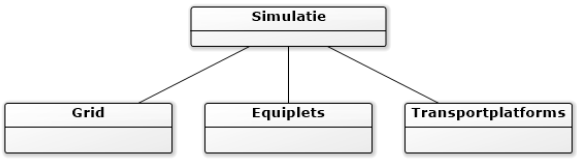

*Figuur 4*

#### 4.1.2.1 De gebruikersinterface

In het subsystem ‘Gebruikersinterface’ kunnen verschillende onderdelen worden onderscheiden. ‘Gebruikersinterface’ bestaat onder andere uit een mogelijkheid van een tool om testbestanden te maken. Dit zijn bestanden waarmee de simulatie zal draaien. Het zal resultaatbestanden moeten kunnen genereren.

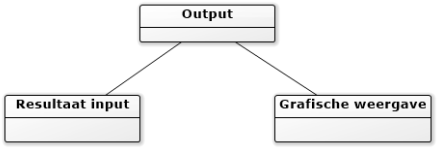


#### 4.1.2.2 De simulatie




Het subsysteem 'Simulatie' maakt gebruik van testbestanden om de simulatie te draaien. Het testbestand bevat een (bepaald) aantal producten die gefabriceerd moeten worden. Om deze producten te fabriceren zijn per product de productiestappen gedefinieerd. De simulatie bestaat uit een productiegrid waarbinnen transportplatforms zich heen en weer bewegen tussen Equiplets. Het productiegrid wordt gesimuleerd door een graph, waarbij de productiestappen uitgevoerd worden door Equiplets die worden vertegenwoordigd in de nodes van de graph. De transportplatforms zullen software-agents zijn die de in het testbestand per product opgeslagen fabricagestappen uitvoeren, door zich van node naar node te verplaatsen.

#### 4.1.2.3 De output




Het subsysteem 'Output' maakt onderscheid in twee onderdelen. Namelijk een 'Resultaat input' en een 'Grafische weergave'. De 'Resultaat input' heeft slechts de functie om de door de simulatie gegenereerde gegevens in te laden. Het onderdeel 'Grafische weergave' zorgt ervoor dat de gegenereerde resultaten logisch en overzichtelijk met elkaar vergeleken worden en worden getoond.

## 4.2 Kwantificatie

In deze sectie wordt het systeem gekwantificeerd, dit wordt gedaan door alle functionele eisen op een rij te zetten in een keydriver graaf, dit gebeurd in paragraaf 4.2.2. Maar eerst worden door middel van het opstellen van een MoSCoW in paragraaf 4.2.1 de functional requirement (must) en de wensen (should) als eis en wensen inzichtelijk gemaakt.

### 4.2.1 MoSCoW

**Must haves**

- Resultaten genereren die uniform zijn, zodat deze later te vergelijken zijn
- Keuze uit centrale en decentrale pathplanning

**Should haves**

- Keuze uit verschillende grid layouts

**Could haves**

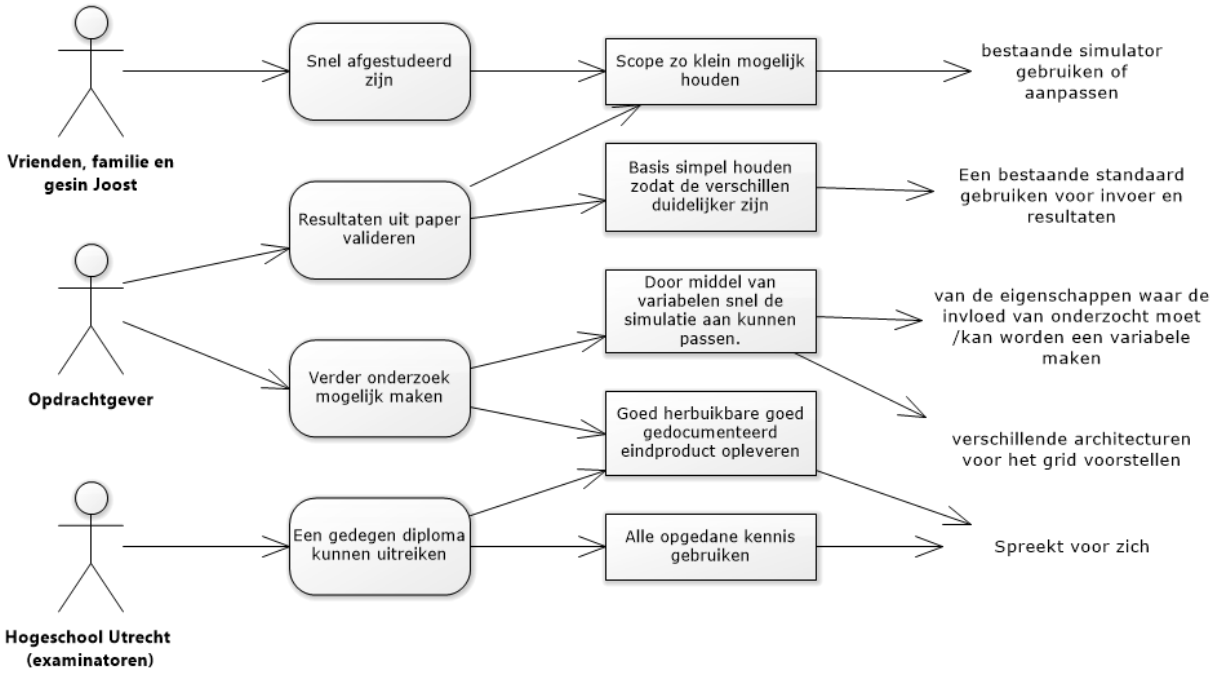
- Functionaliteiten van de Equiplets aanpassen.

**Wouldn't haves**

- Zeer geavanceerde/uitgebreide grafische interface

### 4.2.2 Key driver graaf

Om in de key driver graaf te voldoen aan alle eisen van de stakeholders, kan op basis van de customer objectives view en de application view een lijst wensen worden samengesteld. Aan deze wensen zijn eisen verbonden, deze zijn te zien in d. Hieronder zijn de eisen, die te zien zijn in de key driver graaf, uitgewerkt.



# 5 CONCEPTUAL VIEW

Om te begrijpen welke specificatie het product implementeert, word in dit hoofdstuk een concept uitgewerkt. Dit concept is gebaseerd op de ideeën die beschreven staan in de proefschriften van Leo van Moergestel en Daniel Telgen (Telgen, 2017; van Moergestel, 2014).

## 5.1 Constructie decompositie

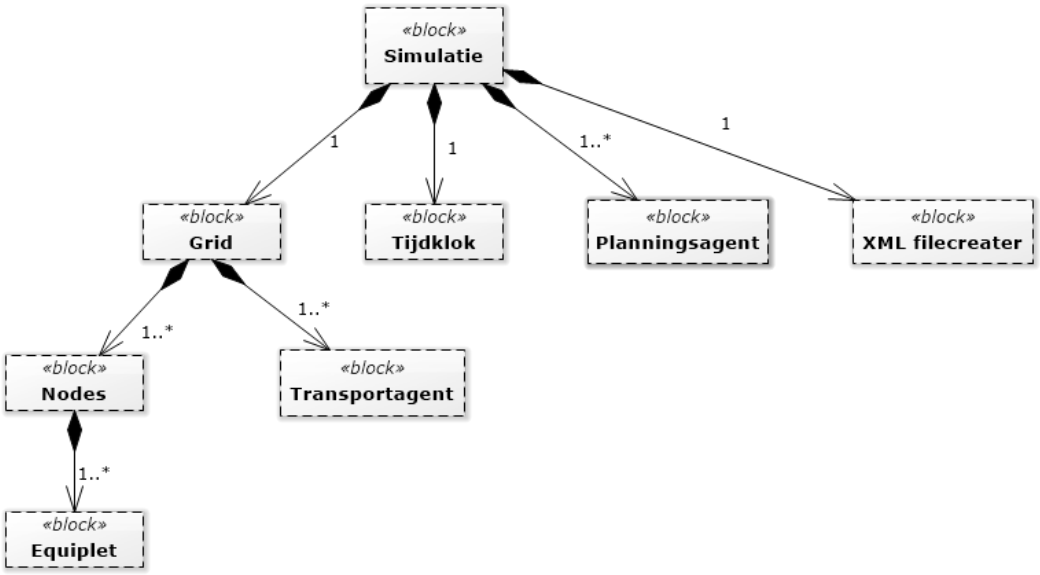
Bij dit concept gaat het om de simulatie. Deze simulatie heeft de volgende onderdelen een grid, nodes, meerdere Equiplets (productie agents), een of meerdere transport agents, planningsagent, een tijdklok.

Als gekeken wordt naar het grid bevat deze nodes, deze nodes zijn plekken waar een transportagent stil mag staan. Ook zijn er vijfentwintig nodes die een Equiplet hebben. Het grid heeft doel om als omgeving te fungeren waarbinnen de Equiplets geplaatst zijn en waarbinnen de transportagents zich dienen te verplaatsen. Vervolgens zijn er de Equiplets, deze hebben de mogelijkheden om productiestappen uit te voeren. Iedere Equiplet is uniek en kan een of meerdere soorten productiestappen uitvoeren. Er zijn een of meerdere transport agents, dit zijn “platforms” die producten tussen de verschillende nodes transporteren. Binnen de simulatie is er een planningsagent, deze kan in een als individuele agent in het transportplatform geïntegreerd zijn of als overkoepelende entiteit werken. Als laatste is er een tijdklok die een tijdseenheid bijhoudt.

Ook bevat de simulatie een XML filecreater die XML-bestanden kan genereren die als lijst fungeert waarmee de producten gemaakt kunnen worden. Het format van dit bestand is door Leo van Moergestel al vastgesteld.

## 5.2 Beslismatrix

Omdat er voor dit project niet meerdere concepten zijn uitgewerkt is het maken van een beslismatrix niet nodig. Mogelijke nieuwe inzichten die gaandeweg het project ontstaan zullen in overleg met Leo van Moergestel al dan niet worden geïmplementeerd.



# 6 REALISATION VIEW

Voor de realisatie is een ontwerp gemaakt gebaseerd op het concept uit hoofdstuk 5. Op basis van dit ontwerp is gekeken wat de beste realisatie is voor het project.

Omdat het binnen dit project om een simulatie gaat waarbij de resultaten betrouwbaar moeten zijn en goed vergelijkbaar moeten zijn is er besloten om niet heel de simulatie zelf te programmeren. Wat ook heeft meegespeeld is dat het volledig zelf maken van een simulatie veel tijd kost om goed te ontwikkelen en te testen, bovendien heeft dit een hoog “opnieuw het wiel uitvinden” gehalte. Er zal in dit hoofdstuk gekeken worden naar een aantal verschillende bestaande simulatietools onderzoek gedaan worden en zal er een worden gekozen. De beslissing voor de keuze van een bestaande simulatietool worden ook bekrachtigd door het advies vanuit de Hogeschool Utrecht (Aldewereld, 2020).

De volgende simulatietools zijn bekeken:

- RePast Simphony
- MESA
- Simulink
- Excel met Analytic Solver Simulation
- RinSIm
- NetLogo

## 6.1 RePast Simphony

Repast Simphony (Argonne National Laboratory, 2020) is een op agenttechnologie gebaseerde modelleringstoolkit en is een platformonafhankelijk Java-gebaseerd modelleringssysteem dat draait onder Microsoft Windows, Apple macOS en Linux. Repast ondersteunt de ontwikkeling van uiterst flexibele modellen van interacterende agents voor gebruik op krachtige computers en computerclusters. Repast Simphony-modellen kunnen in verschillende vormen worden ontwikkeld, waaronder het ReLogo-dialect van Logo (Ozik, Collier, Murphy, & North, 2013), point-and-click statecharts, Groovy of Java, die allemaal vloeiend in elkaar kunnen worden verweven. De variant van Repast die geschikt is voor supercomputers maakt het ook mogelijk C++ te gebruiken in meer of mindere maten. De manier van programmeren is er vooral gericht op Rapid prototyping waarbij de lage leercurve belangrijker is en een degelijke programmeeromgeving (IDE) minder van belang lijkt te zijn.

## 6.2 MESA

Ook MESA (Kazil et al., 2020) is een agent-gebaseerd modelleer toolkit, echter dan in Python. Het stelt gebruikers in staat snel agent-gebaseerde modellen te creëren met behulp van ingebouwde kerncomponenten (zoals ruimtelijke grids en agentplanners) of aangepaste implementaties. Deze zijn te visualiseren met behulp van een (internet)browser-interface en de resultaten zijn te analyseren met behulp van de gegevensanalysetools van Python. MESA is gemaakt om op basis van Python 3 een tegenhanger te zijn van NetLogo, Repast of MASON. MESA wordt veelal gebruikt om simulaties te maken van sociale en ruimtelijke processen, het is daarom minder geschikt voor logistieke problemen (Dean et al., 2000).

## 6.3 Simulink

Simulink (The MathWorks Inc, 2020b) is een betaalde simulatietool die is gemaakt om een systeem te ontwerpen en te simuleren voordat overgegaan wordt op hardware. Het geeft de mogelijkheid ontwerpen te verkennen en te implementeren die anders misschien niet zou worden overwegen, zonder C, C++ of andere “hardware description language” te hoeven schrijven. Alle door Simulink gegenereerde code kan echter wel aangepast worden. Omdat dit niet echt het idee is binnen dit project valt deze eigenlijk verder al af, en zal voor dit project niet dieper in deze tool gedoken worden. Het voordeel is wel dat het heel goed gecombineerd kan worden met MatLab (The MathWorks Inc, 2020a), wat de mogelijkheden erg vergroot.

## 6.4 Analytic Solver Simulation

Solvers, of optimizers, zijn softwaretools die helpen de beste manier te vinden om schaarse middelen toe te wijzen. De resources kunnen grondstoffen, machinetijd of mensen tijd, geld of iets anders in beperkte voorraad zijn. Dit past dus heel goed in de ideologie van Lean en Agile produceren. De "beste" of optimale oplossing kan het maximaliseren van de winst, het minimaliseren van de kosten of het bereiken van de best mogelijke kwaliteit betekenen. Op deze manier kan een bijna oneindige verscheidenheid aan problemen worden aangepakt. Deze solver werkt als extension op Excel van Microsoft, aangezien deze oplossing wel heel ver afwijkt van het opleveren van een “software” simulatie valt deze tool ook af als oplossing binnen deze opdracht (Frontline Systems Inc, 2020).

## 6.5 RinSim

RinSim (van Lon, 2018) is een uitbreidbare logistieke simulator met ondersteuning voor (de) gecentraliseerde algoritmen voor ophaal- en bezorgproblemen en (Automatisch geleide voertuigen) AGV-routing. De simulator is gericht op eenvoud en consistentie, waardoor het ideaal is voor het uitvoeren van wetenschappelijke simulaties. Verder heeft bij het ontwerpen van de software de softwarekwaliteit bovenaan gestaan, wat resulteert in een steeds beter wordende testsuite en documentatie. Wat het ook een erg interessant softwarepakket maakt is dat het specifiek is ontwikkeld voor wetenschappelijk onderzoek en het daarvoor ook wordt gebruikt binnen de Distrinet Research Group van de KU Leuven (van Lon & Holvoet, 2012) Een groot voordeel is, is dat RinSim goed geïntegreerd kan worden in IntelliJ IDEA (JetBrains s.r.o, 2020), wat een zeer uitgebreide IDE is voor Java.

## 6.6 NetLogo

NetLogo (Wilensky, 2016) is een multi-agent programmeerbare simulatieomgeving. Ook dit platform wordt door heel veel studenten, docenten en onderzoekers over de hele wereld gebruikt. Het is geschreven in Scala en Java en maakt gebruik van de programmeertaal Logo. Daarom klinkt het in mijn ogen minder geschikt voor dit project. Bij NetLogo lijkt net als bij RePast de leercurve belangrijker dan de uitgebreide programmeeromgeving.

## 6.7 Beslismatrix

Tabel 1 laat de beslismatrix zien, met per tool de eigenschappen, voordelen en nadelen.

*Tabel 5 Beslismatrix simulatietools*

| Tool | Eigenschappen | Voordelen | Nadelen |
|---|---|---|---|
| **RePast Simphony** | Zeer uitgebreid<br>C++<br>Java<br>Open-Source | Verschillende programmeertalen mogelijk<br>Lage leercurve | Veel verschillende versies van de tool<br>Naar mijn mening te veel functionaliteit |
| **MESA** | Modulaire componenten<br>Browser-gebaseerde visualisatie<br>Ingebouwde tools voor analyse<br>Voorbeeld modelbibliotheek | Ingebouwde analyse tools<br>Python | Browser-based visualisatie<br>Meer gericht op “sociale” simulaties |
| **Simulink** | Van dezelfde makers als MATLAB | Genereert zelf veel code<br>Goed te combineren met MATLAB | Betaald<br>Closed source<br>Automatisch gegenereerde code soms lastig te volgen (persoonlijke ervaring) |
| **Excel met Analytic Solver Simulation** | Extensie voor Excel | Kan zeer complexe problemen oplossen | Betaald |
| **RinSIm** | Java (Maven)<br>Ondersteuning voor het verzamelen van statistieken.<br>Ondersteuning voor gecentraliseerde en gedecentraliseerde algoritmen. | Grondig getest<br>Goed gedocumenteerd<br>Nadruk op wetenschappelijke correctheid<br>Modulair<br>Configureerbaar<br>Open source<br>IntelliJ IDEA | Voor het laatst in 2018 bijgewerkt |
| **NetLogo** | Java<br>Scala<br>Te Programmeren met Logo | Wereldwijd gebruikt.<br>Lage leercurve | Logo programmeertaal |

## 6.8 Conclusie

Nadat al deze simulatietools zijn vergeleken met elkaar gaat de voorkeur uit naar het gebruik van RinSim. Hetgeen wat de doorslag heeft gegeven is dat deze tool is ontwikkeld op universitair niveau en er bij de ontwikkeling centraal heeft gestaan om wetenschappelijk correcte gegevens te genereren. Verder heeft het een voordeel dat het is geschreven in JAVA en het volledig open-source en goed gedocumenteerd is.

Mijns inziens is dit dus de beste keuze. Zeker gezien de beperkte tijd is de keuze van een goede bestaande tool beter dan een volledig op de klant gericht stuk software zelf te maken. Hierdoor kan het echter wel zijn dat sommige implementaties anders gedaan moeten worden omdat dit beter binnen RinSim passen.

Het is dus best mogelijk dat sommige onderdelen buiten RinSim worden gemaakt, zoals bijvoorbeeld de software die is bedoeld voor het genereren van testbestanden.

# 7 BRONNEN


Aldewereld, H. M. (2020, 8 april 2020). [Email conversation].

Argonne National Laboratory. (2020, Oct 5 2020). Repast Simphony Reference Manual. Retrieved from https://repast.github.io/docs/RepastReference/RepastReference.html#_introduction

Dean, J. S., Gumerman, G. J., Epstein, J. M., Axtell, R. L., Swedlund, A. C., Parker, M. T., & McCarroll, S. (2000). Understanding Anasazi culture change through agent-based modeling. *Dynamics in human and primate societies: Agent-based modeling of social and spatial processes*, 179-205.

Frontline Systems Inc. (2020). What's the Easiest Way to Solve Optimization Problems? | solver. Retrieved from https://www.solver.com/whats-easiest-way-solve-optimization-problems

JetBrains s.r.o. (2020). IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. Retrieved from https://www.jetbrains.com/idea/

Kazil, J., Pike, T., Masad, D., Garson, J., Mitchell, D. J., Hoover, A., . . . Ells, K. (2020, Jul 24, 2020). Mesa: Agent-based modeling in Python 3+. Retrieved from https://mesa.readthedocs.io/en/master/index.html

Muller, G. (2004). CAFCR: A multi-view method for embedded systems architecting; balancing genericity and specificity.

Ozik, J., Collier, N. T., Murphy, J. T., & North, M. J. (2013). *The ReLogo agent-based modeling language.* Paper presented at the 2013 Winter Simulations Conference (WSC).

Telgen, D. (2017). *Grid Manufacturing.* (PHD), Utrecht University, Utrecht.

The MathWorks Inc. (2020a). MATLAB - MathWorks - MATLAB & Simulink. Retrieved from https://nl.mathworks.com/products/matlab.html

The MathWorks Inc. (2020b). Simulink - Simulation and Model-Based Design - MATLAB & Simulink. Retrieved from https://nl.mathworks.com/products/simulink.html

van Lon, R. R. (2018, 11-06-2020). RinSim | RinSim. Retrieved from https://rinsim.rinde.nl/

van Lon, R. R., & Holvoet, T. (2012). *RinSim: A simulator for collective adaptive systems in transportation and logistics.* Paper presented at the 2012 IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems.

van Moergestel, L. (2014). *Agent Technology in Agile Multiparallel Manufacturing and Product Support.* Utrecht University, Retrieved from https://dspace.library.uu.nl/handle/1874/298812

van Moergestel, L. (2020, 26-05-2020). [Telefonisch overleg 26-05-2020].

van Moergestel, L., Puik, E., & Meyer, J.-J. (2018, 24 - 28, 2018). *A Software Architecture for Transport in a Production Grid.* Paper presented at the INTELLI 2018, Venice, Italy

Wilensky, U. (2016). NetLogo Home Page. Retrieved from https://ccl.northwestern.edu/netlogo/

# II. Source Code

De source code en eindproducten zijn te vinden op mijn GitHub. Deze is te vinden op:
https://github.com/Josque/Afstudeerproducten.git