|  |
| --- |
|  |
| P2P Containing 2014/2015 |
| Groep 8 |
|  |
| **Thom Wouda**  **Bram Honingh**  **Moreno Bralts**  **Matthijs Hovestad**  **Jeroen van der Meulen** |
| **8-1-2015** |

**Samenvatting:**

Voor ons project van het tweede jaar informatica moesten we de haven van Rotterdam naar maken. Deze opdracht werd door 8 groepen behandeld en wij waren groep 8.

Voor de analyse is er een usecase diagram gemaakt, hierdoor waren de richtlijnen duidelijk te volgen en zijn eventuele problemen voorkomen.

De opleiding heeft voor een planning gezorgd die de groep zonder moeite kon volgen. Echter hebben wij deze planning wel deels aangepast.

Rondom het programmeren hebben wij wat afspraken gemaakt over het testen van de gemaakte code. Eerdere pull request die methodes toevoegt moet gepaard zijn met een overeenkomstige unit test. De unit tests van de 3D interface kunnen mogelijk overgeslagen worden, en vervolgens vervangen worden met UX testing. De programmalogica moet echter wél unit tests hebben.

Alle pull requests en directe commits worden getest via een Continuous Integration server. Hiervoor wordt Travis-CI gebruikt (vooral bekend in de Ruby community). De CI server blijft stil totdat er een commit gepusht wordt die niet compileert en/of niet goed uit de tests komt. Het doel is om geen pull requests te accepteren waarover de CI server zeurt. Continuous Integration is alleen maar mogelijk in de controller.

De eindverantwoordelijke voor tests en QA is Moreno.

Uiteindelijk hebben we er voor gezorgd dat de applicaties aan alle eisen van school voldoen. Wij zijn erg blij met het resultaat dat is neergezet.

**Voorwoord:**

Na een periode bezig te zijn geweest met de Containing applicatie heeft onze groep dit met veel plezier afgerond. Wij zijn zeer tevreden over ons eindresultaat.

Tijdens deze opdracht hebben we veel contact gehad met onze tutor Jos Foppele. Ook hebben wij regelmatig gesproken met Wouter van der Ploeg en Dick Bruin. Zij hebben ons veel geholpen tijdens het project daarom willen wij hun ook graag bedanken.

Veel plezier met het lezen van ons verslag,

Groep 8

**Verklarende woordenlijst:**

Sequence diagram

Een sequence diagram geeft weer hoe verschillende processen met elkaar omgaan en in welke volgorde dit gebeurd.

Class diagram

Een class diagram laat de structuur zien van een programma door middel van de gebruikte classes.

Statetransition diagram

Een State transition diagram geeft de verschillende mogelijke toestanden van een systeem weer.

Usecase diagram

Usecase diagrammen worden voornamelijk gebruikt in software engineering om de eisen van een systeem in kaart te brengen. Het doel van een usecase diagram is om een grafisch overzicht te geven van de functionaliteiten van een systeem.

Datamodel

Het ontwerp van hoe de database er uit komt te zien.

Database

Een grote hoeveelheid informatie die elektronisch is opgeslagen. Deze informatie is nodig om het programma goed te laten werken.

Threading

Threading is het uitvoeren van meerdere taken tegelijkertijd, buiten java ook gekend onder de benaming multitasking.

Maven

Maven is een softwaregereedschap voor Java-projectmanagement en geautomatiseerde softwarebouw.

GitHub

GitHub is een website die gebruikmaakt van Git om in groepen aan softwareontwikkeling te doen.

Inhoud

[1. Inleiding: 5](#_Toc408488229)

[2. Opdrachtomschrijving: 5](#_Toc408488230)

[3. Analyse: 6](#_Toc408488231)

[4. Architectuur: 7](#_Toc408488232)

[4.1 Simulatie 7](#_Toc408488233)

[4.2 Controller 7](#_Toc408488234)

[4.3 Webapp 7](#_Toc408488235)

[5. Aanpak en taakverdeling: 8](#_Toc408488236)

[6. Problemen en oplossingen: 9](#_Toc408488237)

[7. Conclusie: 9](#_Toc408488238)

[8. Evaluatie: 9](#_Toc408488239)

[Bijlageformulier: 10](#_Toc408488240)

[I. Logboeken: 10](#_Toc408488241)

[II. Diagrammen: 10](#_Toc408488242)

[III. Planning: 16](#_Toc408488243)

# 1. Inleiding:

Dit verslag is geschreven naar aanleiding van P2P Informatica 2014 aan de NHL Hogeschool te Leeuwarden.

Voor dit project was er vanuit de opleiding een opdracht verstrekt. Ook waren de groepen al van tevoren ingedeeld waardoor het voor iedereen nog een verrassing was met wie je samen zou gaan werken.

In dit verslag kunt u o.a. lezen wat onze opdracht was, hoe dit is aangepakt, wat voor problemen de groep tegen is gekomen en wat het uiteindelijke resultaat is.

# 2. Opdrachtomschrijving:

De opdracht voor dit project was het maken van een simulatie van de haven van Rotterdam. Om dit te realiseren moesten er drie onafhankelijke programma’s geschreven worden: een simulator, een controller en een applicatie die op een mobiel apparaat gebruikt kan worden.

De simulator moet ervoor zorgen dat er een 3D weergave komt van de haven. Dit houd in dat het hele terrein van de haven te zien is, inclusief de containers, bewegende kranen en AGV’s. De simulator moet geschreven worden in Java.

De controller moet door middel van meerdere XML bestanden de simulator gaan aansturen. De controller moet ervoor zorgen dat de containers naar de juiste locaties binnen het platform worden gebracht , en ook weer op de juiste tijd het platform verlaten. De controller moet geschreven worden in Java.

Met de applicatie moet het mogelijk zijn om op een mobiel apparaat actuele informatie te zien over de simulatie. Hierop kan bijvoorbeeld diagrammen weergeven worden met het aantal containers. De applicatie moet draadloos gebruikt kunnen worden en werken op Android of iOS.

# 3. Analyse:

Voor de analyse heeft onze groep gekozen om een usecase diagram te maken met bijbehorende klasse- en statetransition diagram (Bijlage II Figuur 1, 2, 3, 4 en 5). Er is hiervoor gekozen omdat er aan het begin van dit project nog niet genoeg kennis en inzicht van het containing project was. Op deze manier konden alle functionaliteiten van beide applicaties goed in beeld worden gebracht.

Na het maken van de uml zijn er verschillende schetsen gemaakt voor de opzet in de simulatie en het communicatie systeem. Hierdoor was het voor ons duidelijk te zien hoe applicatie eruit komt te zien en hoe wij een solide communicatie tussen de simulatie en de controller behouden (Bijlage II Figuur 6 ).

Ook is uitgeschreven hoe de communicatie voor beide applicaties threading gebruiken zodat deze kunnen blijven luisteren naar elkaar en ondertussen wat kunnen verzenden, deze schets is oorspronkelijk voor de controller geschetst maar werkt bij de simulatie op dezelfde manier (Bijlage II Figuur 7 ). Hierbij zijn de grote cirkels met een driehoek in de omtrek van de cirkel een nieuwe thread die wordt aangemaakt en geloopt wordt waarbij de controller thread eigenlijk de main loop van de controller is. ( Die automatisch onderwater door java wordt aangemaakt dmv de main methode aan te gebruiken ).

# 4. Architectuur:

4.1 Simulatie

De simulatie wordt in jME3 (jMonkeyEngine 3) gemaakt. De simulatie is afhankelijk van de controller door middel van signaalverkeer. Het project maakt gebruik van de standaard BasicGame template die geleverd wordt door de jME SDK.

Dit signaalverkeer ziet er ongeveer zo uit:

1. Controller geeft opdracht aan Simulator.
2. Simulator verwerkt de opdracht en stuurt een OK signaal terug.
3. Simulator maakt de opdracht af en stuurt een tweede OK signaal terug.
4. Controller geeft een nieuwe opdracht terug aan de Simulator.

4.2 Controller

De controller is een standaard Java project. Het is afhankelijk van de simulator door middel van signaalverkeer. Er wordt gebruikt gemaakt van Maven als packaging & build tool. Dit zorgt ervoor dat extra libraries makkelijk toegevoegd kunnen worden zodra ze nodig zijn. Er wordt daardoor gekozen voor een flexibele aanpak van 3rd party libraries: Als een library beschikbaar is via Maven en makkelijk geïntegreerd kan worden in het project, dan is het in principe geen probleem om de library toe te voegen als dependency.

Er wordt echter wel sterk benadrukt dat jME3 niet een dependency van de controller kan zijn. Er is geen Maven package voor jME3, en de package bevat alleen maar simulator logica die in de simulator thuishoort.

De controller leest de geleverde XML files in via de SAX XML parser. Er wordt een (bijna) 1 op 1 overzetting gedaan van de data uit de XML file naar een Container-object. Dit dataverkeer heeft één richting: Er kan niet terug naar de XML files geschreven worden.

De controller moet een goede beginstaat hebben die afgeleid kan worden vanuit de XML files. Verder regelt de controller al het verkeer van de voertuigen die gesimuleerd worden. Deze aansturing vloeit primair uit het Dijkstra-algoritme.

4.3 Webapp

Om informatie over de staat van de controller te kunnen visualiseren op een mobiel apparaat, is er besloten om een web-pagina te gebruiken. Vanuit de controller (controller) wordt de data opgeslagen in een database. Deze gegevens worden geladen op de web-pagina.

# 5. Aanpak en taakverdeling:

Na de aftrap van dit project zijn de projectleden bijeen gekomen. Hierbij is er besproken en gedocumenteerd wat er gedaan moet worden. In de eerste week hebben we ons voornamelijk bezig gehouden met discussiëren en plannen van de functionaliteit van het programma. Hiervoor zijn usecase diagrammen, class diagrammen, toestandsdiagrammen en sequence diagrammen gemaakt. Dit voor beide de controller en simulator.

Na het maken van de diagrammen is er overlegt welke manier van opslag er gebruikt zou worden. Hierbij is iedereen akkoord gegaan met GitHub. Hierbij is één persoon aangewezen voor het beheer hiervan. Op deze manier kan iedereen code sturen en zal deze persoon eerst de code controleren voordat het samengevoegd wordt met het project.

Ten slotte is er een manier van communicatie geregeld zodat er ook buiten schooltijden en bij afwezigheid makkelijk met iedereen gecommuniceerd kan worden.

Qua planning heeft de groep besloten om aan de hand van de door school gegeven planning te werken. (Bijlage III Tabel 1). Vervolgens is er afgesproken om elke doordeweekse dag van 10 tot 3 op school met het project bezig te gaan. Als groep vonden we het belangrijk om elke dag samen bijeen te werken zodat er bij problemen spoedig overlegt en/of geholpen kan worden.

Voor de taakverdeling hebben we besloten om voornamelijk de issue tracker van GitHub te gebruiken. Wanneer er een nieuwe taak ontstond werd hier een issue van gemaakt. Wanneer een groepslid klaar is met zijn huidige taak word deze gesloten en kan diegene uit de lijst met issues een nieuwe taak kiezen. Daarnaast is er binnen de groep op verschillende momenten besproken wie welke issue gaat oppakken en wie waarmee bezig is.

6. Problemen en oplossingen:

Hieronder staat een tabel met risico’s die kunnen voorkomen en zijn voorgekomen, zoals ziekte, en hoe wij daarmee om gaan en zijn gegaan.

|  |  |
| --- | --- |
| **Risico** | **Oplossing** |
| Ziekte. | Goed communiceren en als het nodig is planning aanpassen. |
| Slechte communicatie. | Vragen stellen als je vast zit. |
| Opdrachtgever veranderd van gedachten. | Planning aanpassen. |
| Dataverlies. | Gebruik van GitHub. |
| Achterlopen op planning. | Planning aanpassen. |

Tabel 1 – Risk analyse

Halverwege het project kwam Bram in de groep. Omdat Bram niet bekent was met het project was het best lastig om hem taken te geven aangezien het project al in een gevorderd stadium was. Dit hebben we opgelost door hem veel mee te laten kijken en hem zelf ook taken te geven.

Verdere problemen zijn er niet echt opgetreden, behalve dat het teamlid Ruben ervoor koos om met de opleiding te stoppen en daarom de project groep verliet. Dit hebben wij goed en snel opgepakt door alle code die werd beheerd door Ruben over te zetten naar de GitHub van Matthijs zodat we verder konden waar we gebleven waren.

# 7. Conclusie:

Gedurende de afgelopen periode hebben we met onze groep gefocust op alle gestelde eisen van dit project. Dit is dan ook terug te zien in ons eindresultaat.

Ook heeft de groep een aantal van wensen weten te realiseren waaronder het gebruik van een database.

# 8. Evaluatie:

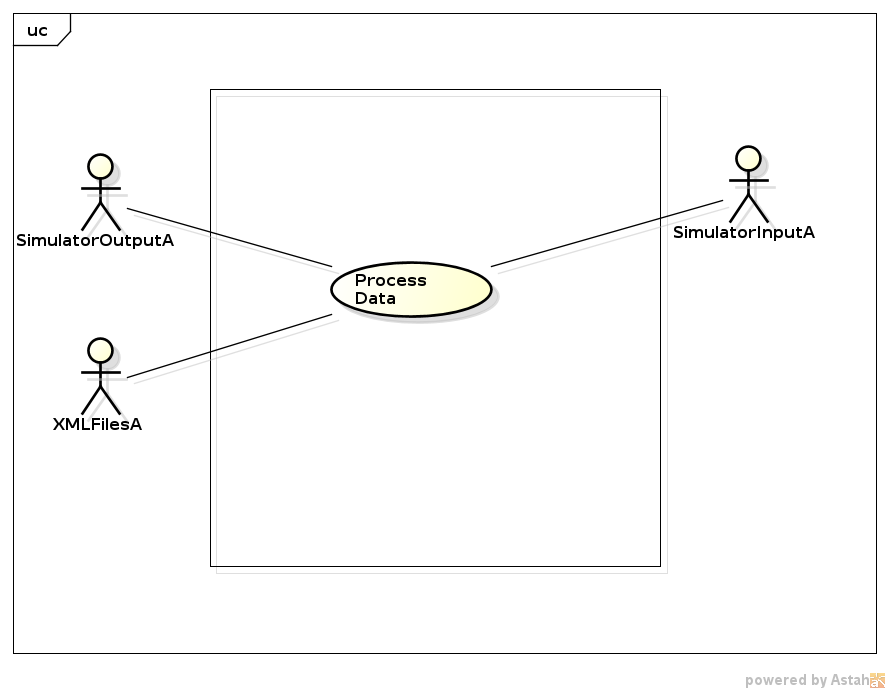
We vonden het een leuke en leerzame opdracht en zijn ook erg blij met het uiteindelijke resultaat. Wij zijn verschillende uitdagingen tegen gekomen qua programmeren in java en qua graphics met jMonkey. Naast het programmeren was het een nieuwe uitdaging om samen met willekeurige mensen in een groepje samen te werken. Bij deze uitdagingen zijn op verschillende momenten situaties ontstaan die we samen als groep hebben opgelost. Van deze ervaringen heeft iedereen geleerd.

# Bijlageformulier:

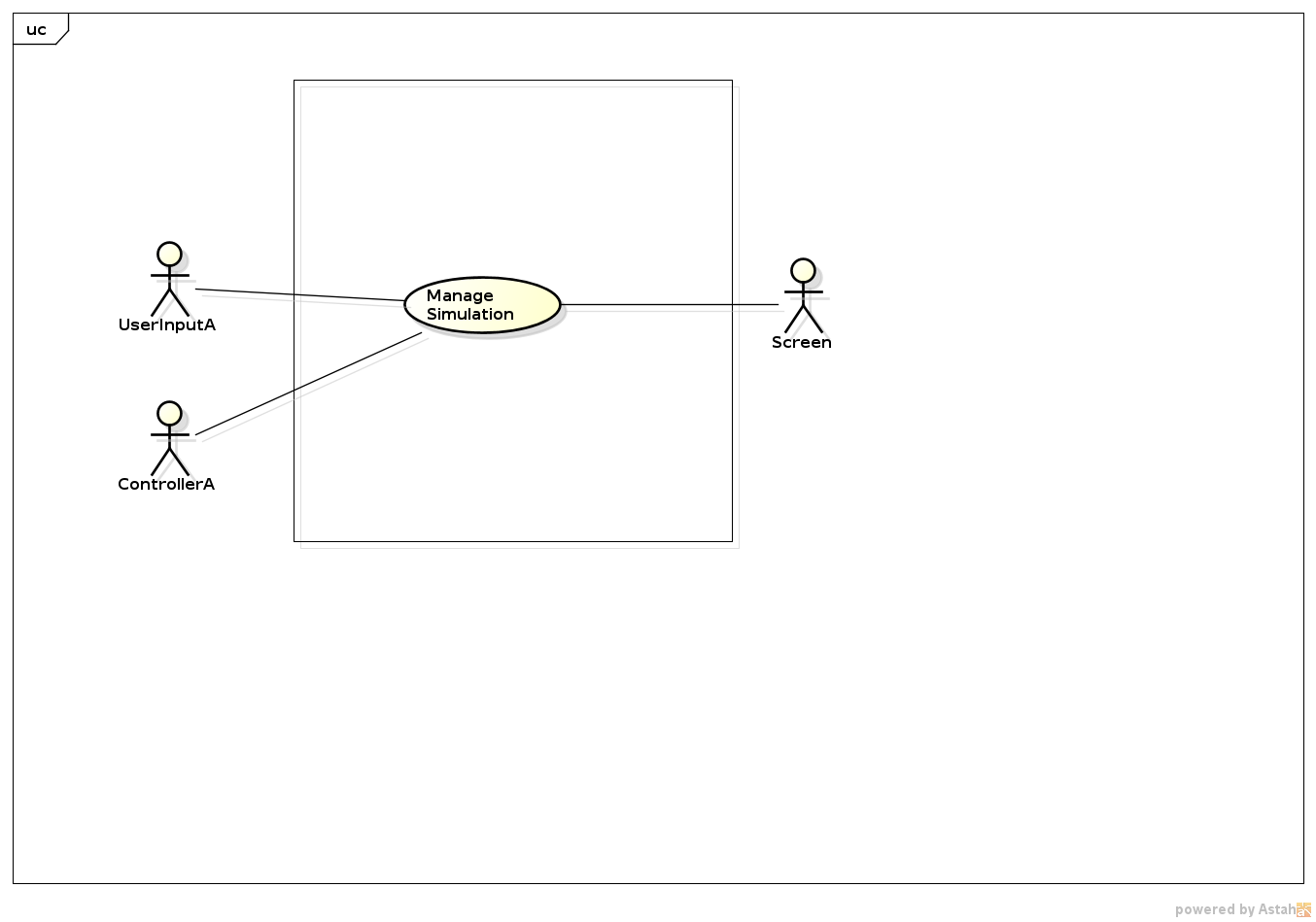
## Logboeken:

Voor ons logboek zie: https://github.com/pasibun/containing/wiki/Notes

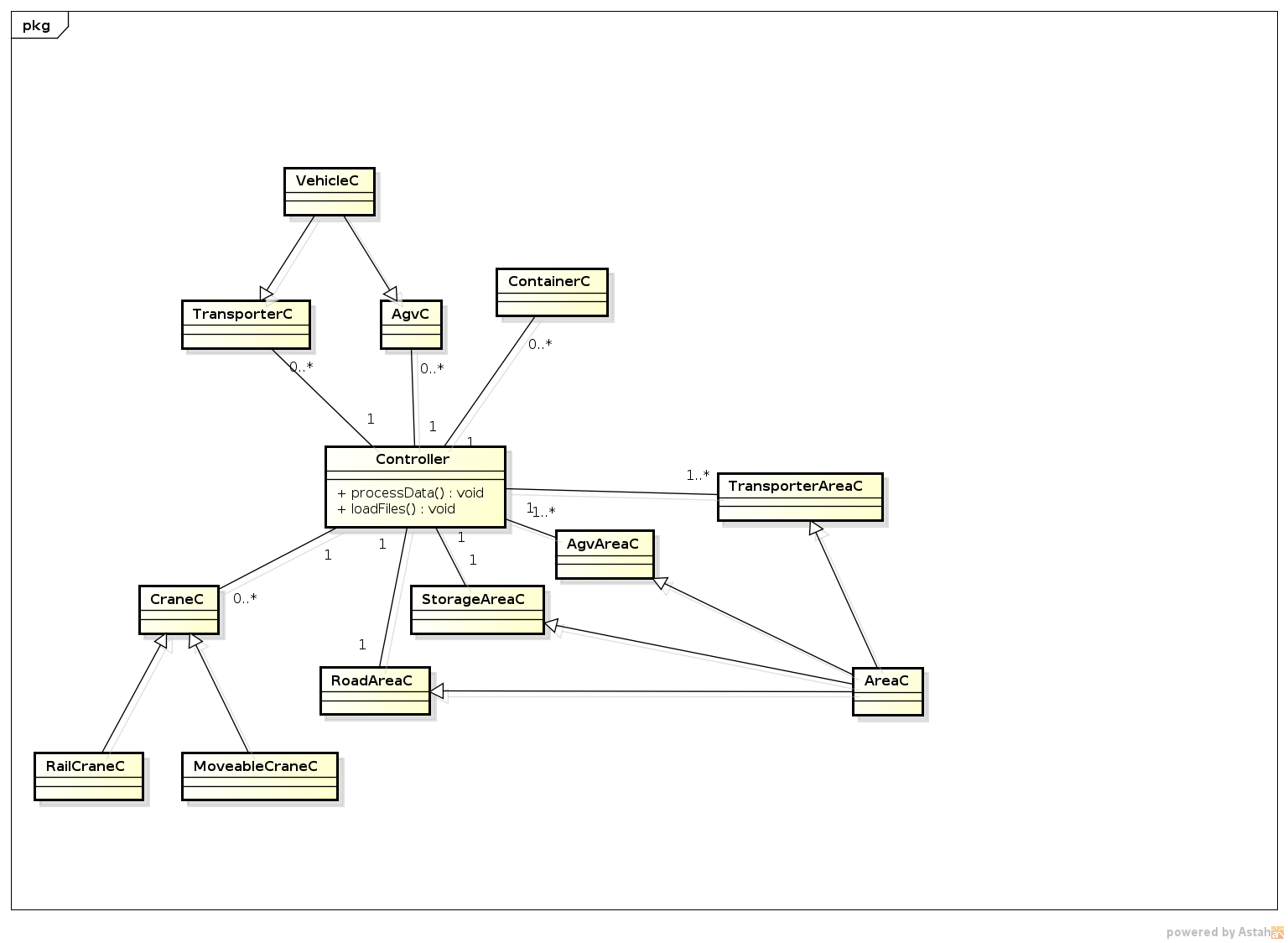
## Diagrammen:



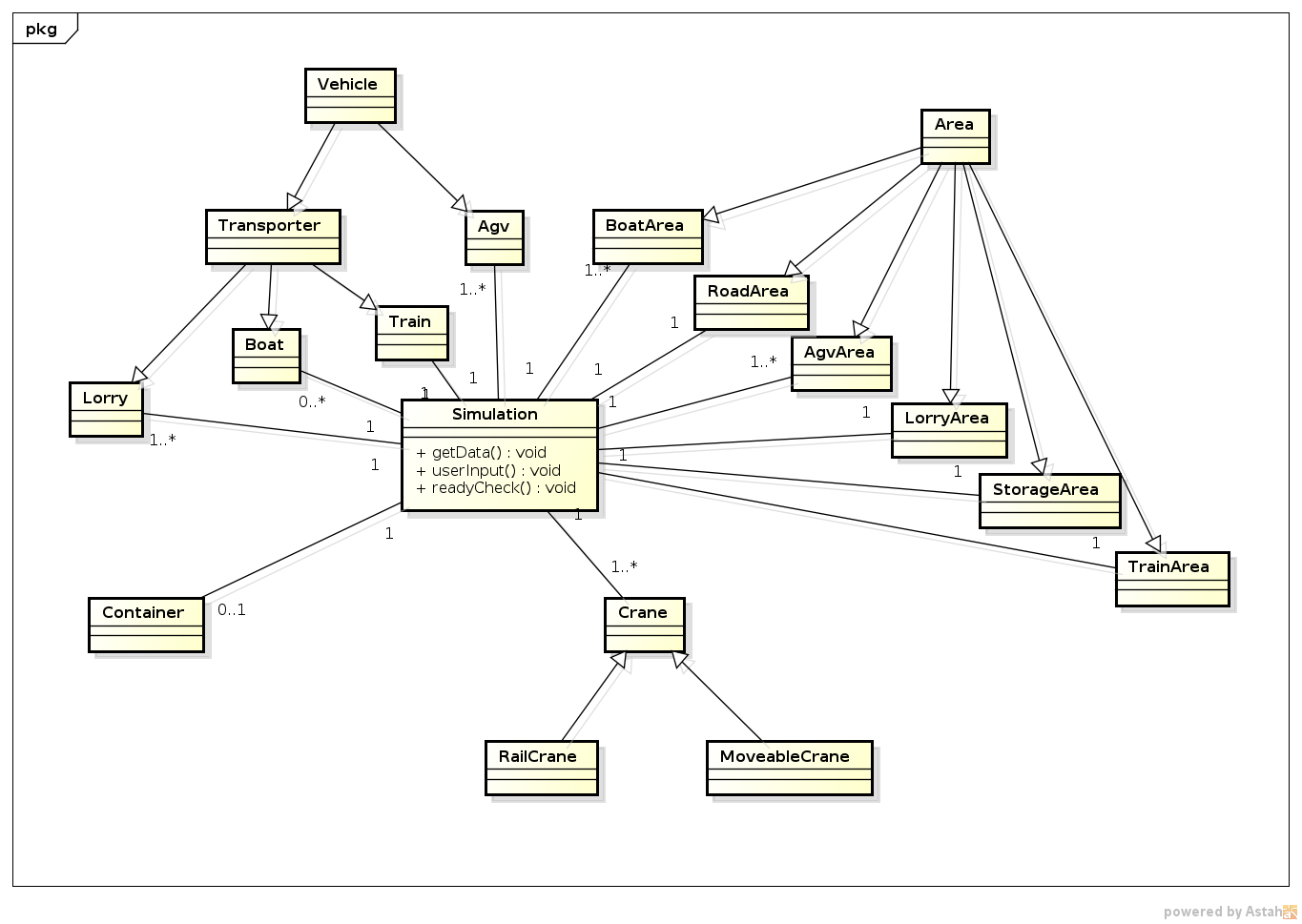
Figuur 1 Usecase Controller



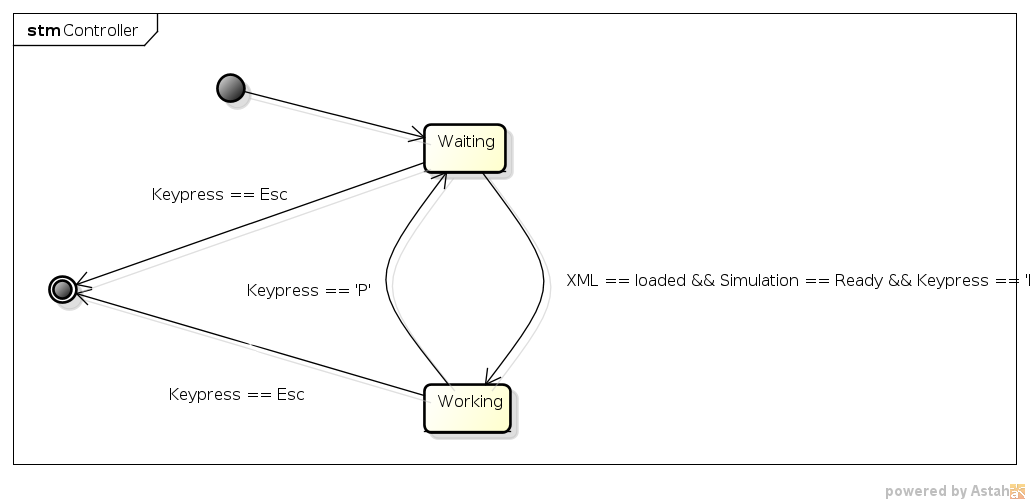
Figuur 2Usecase Simulator



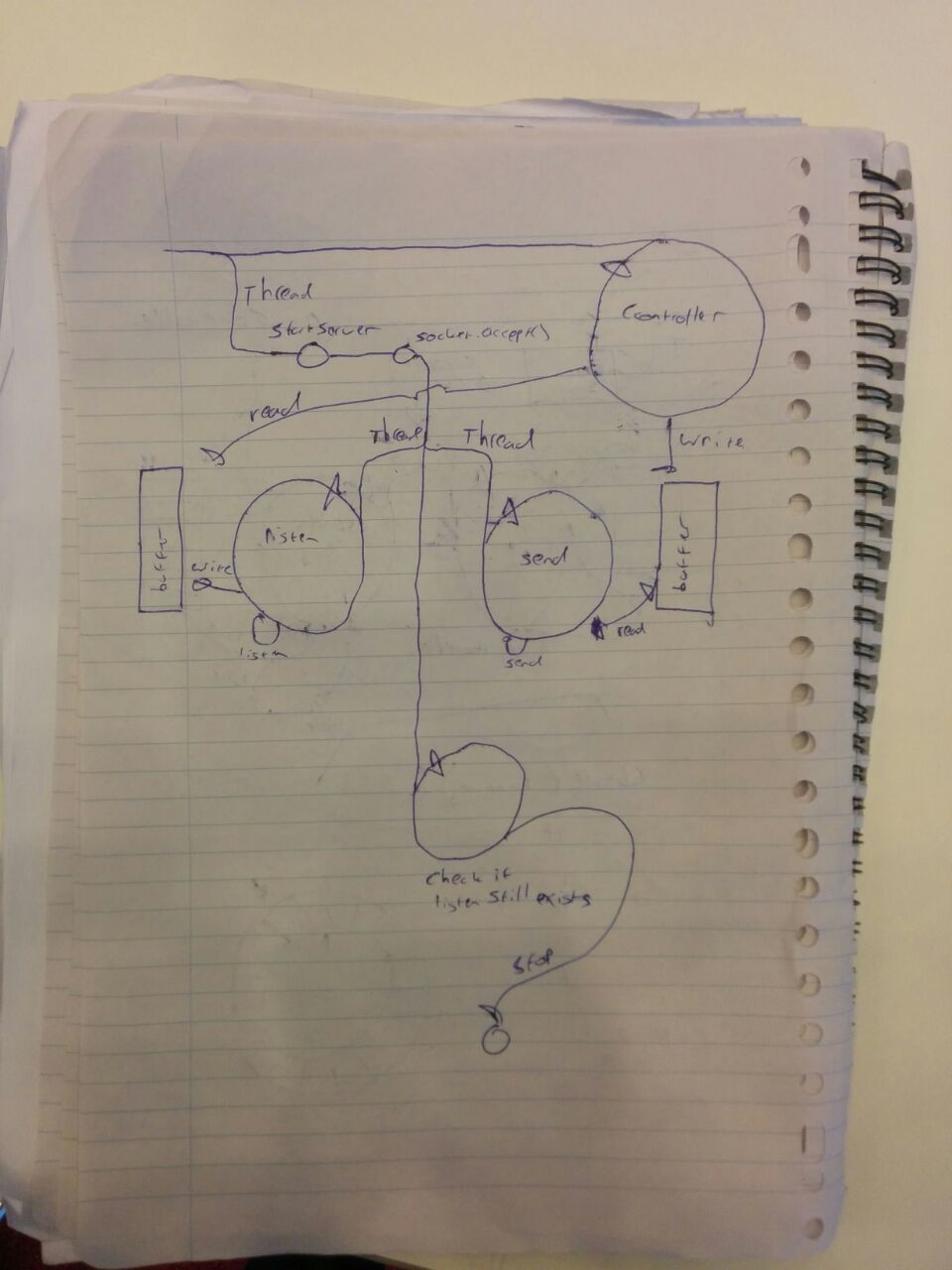
Figuur 3 Klassediagram Controller



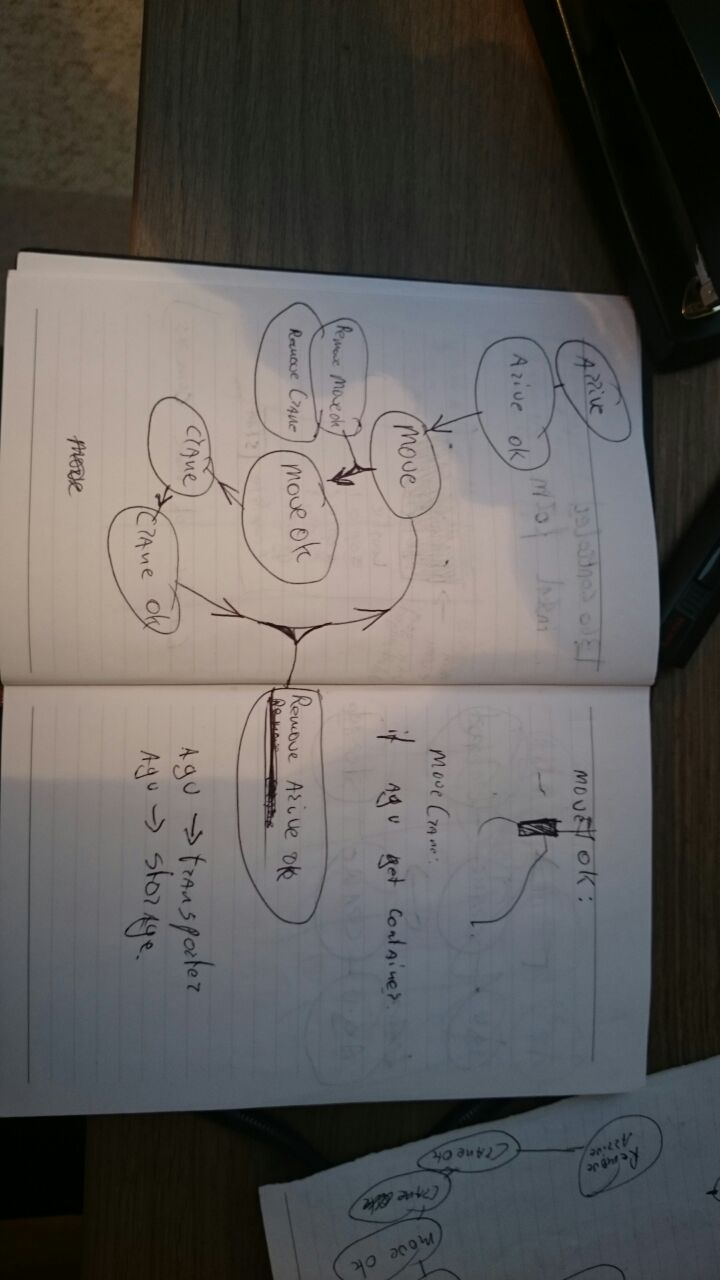
Figuur 4Klassediagram Simulator



Figuur 5 STD Controller & Simulator



Figuur 6Threading Communicatie



Figuur 7 Communicatie Message systeem

## Planning:

Tabel 1

