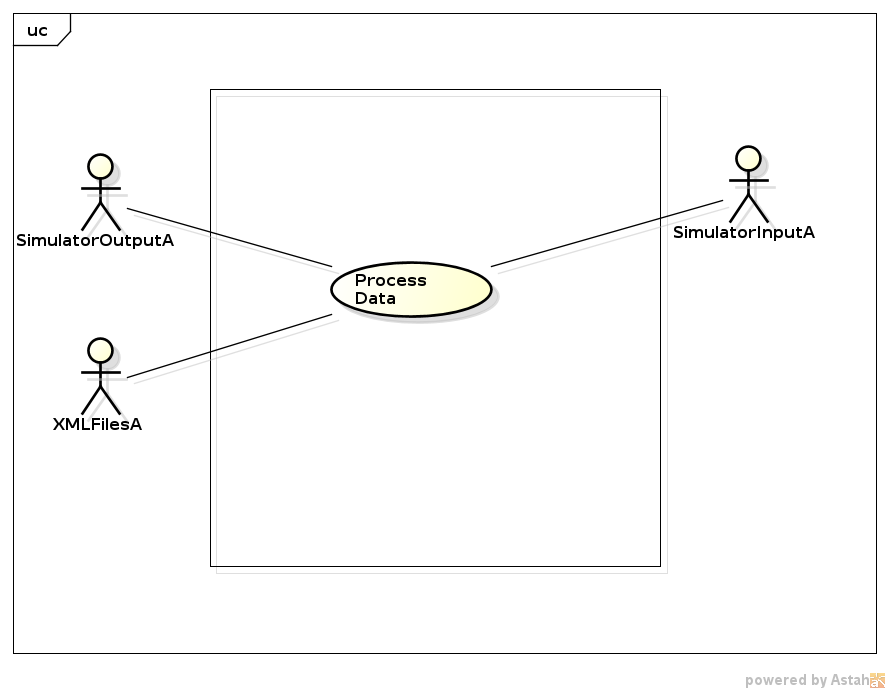
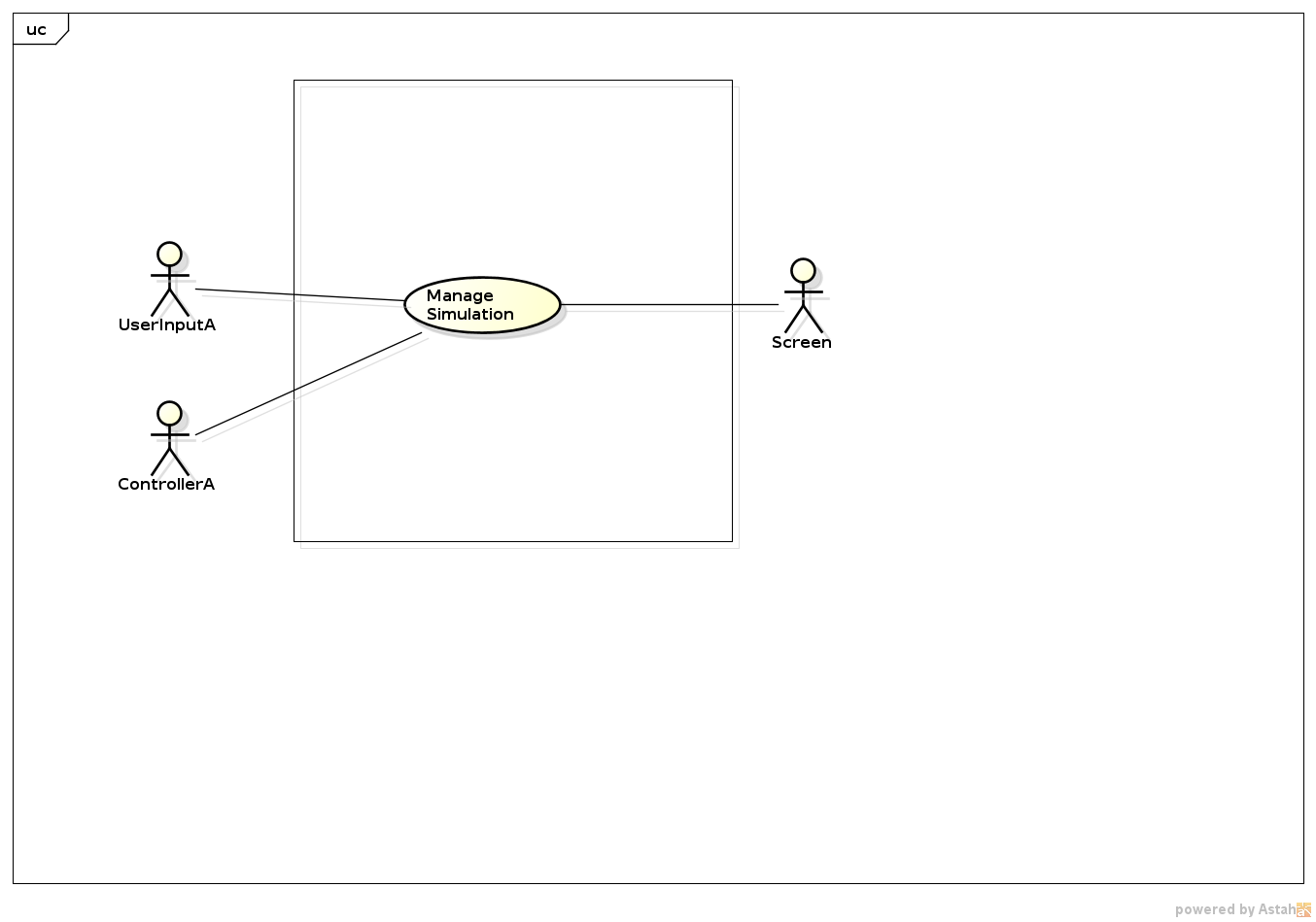
# 3. Analyse:

Voor de analyse heeft onze groep gekozen om een usecase diagram te maken met bijbehorende klasse- en statetransition diagram (Figuur, II, III, IV & V). 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.

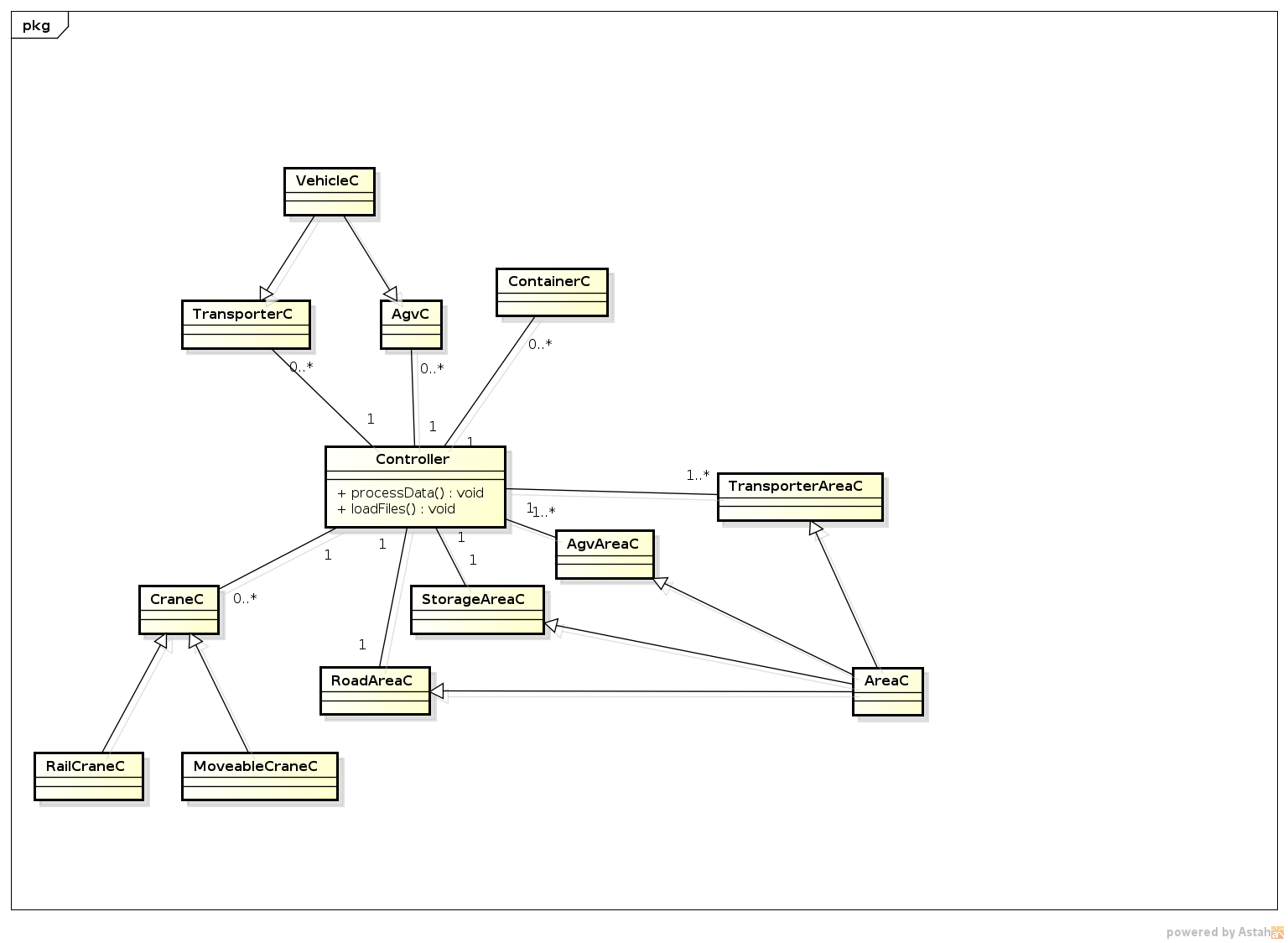
Na het maken van de uml zijn er verschillende schetsen gemaakt voor de opzet in de simulatie en het communicatie systeem. Hierdoor was het voor ons duidelijk te zien hoe applicatie eruit komt te zien en hoe wij een solide communicatie tussen de simulatie en de controller behouden (Figuur VI ).

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 ( Figuur VII ). 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 ). 

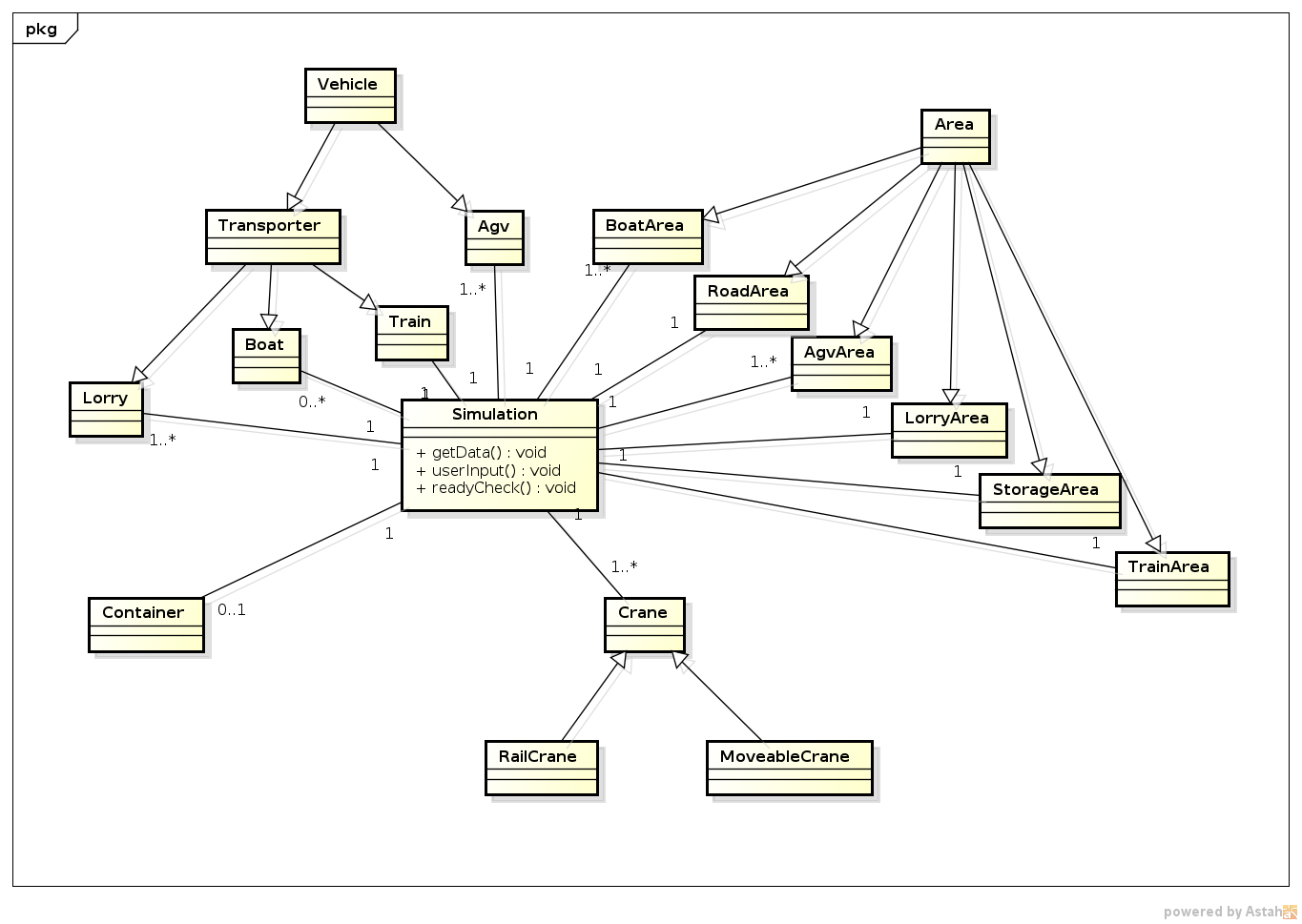
Figuur Usecase Controller



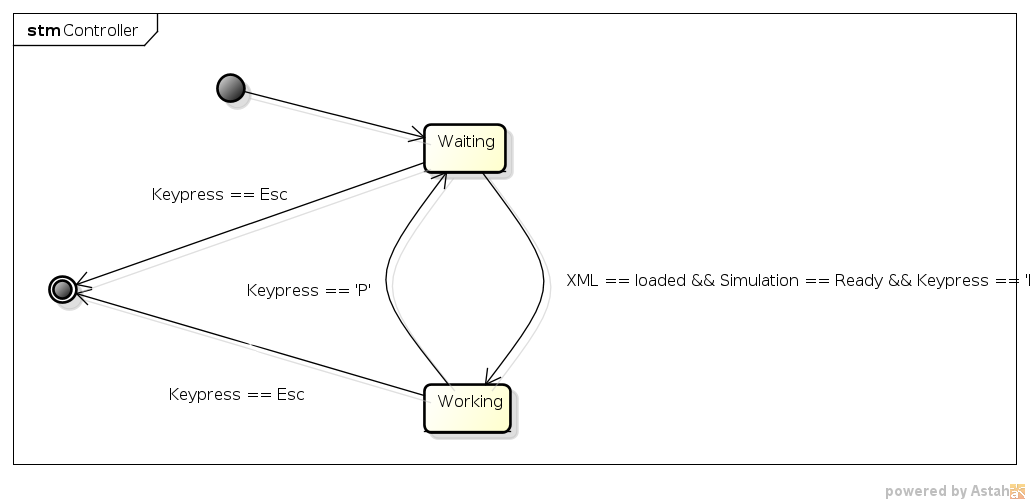
Figuur 2 Usecase Simulator



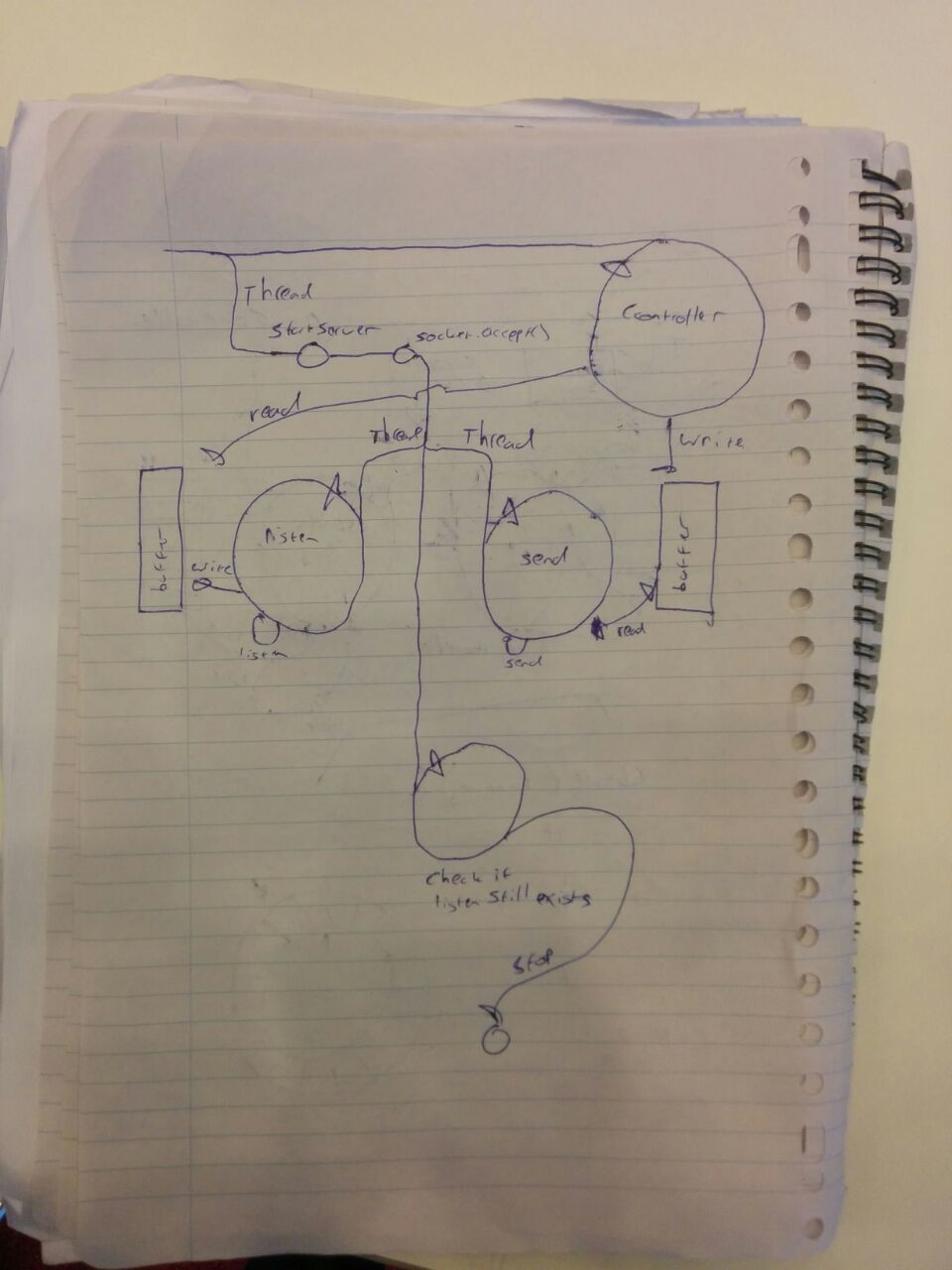
Figuur 3 Klassediagram Controller

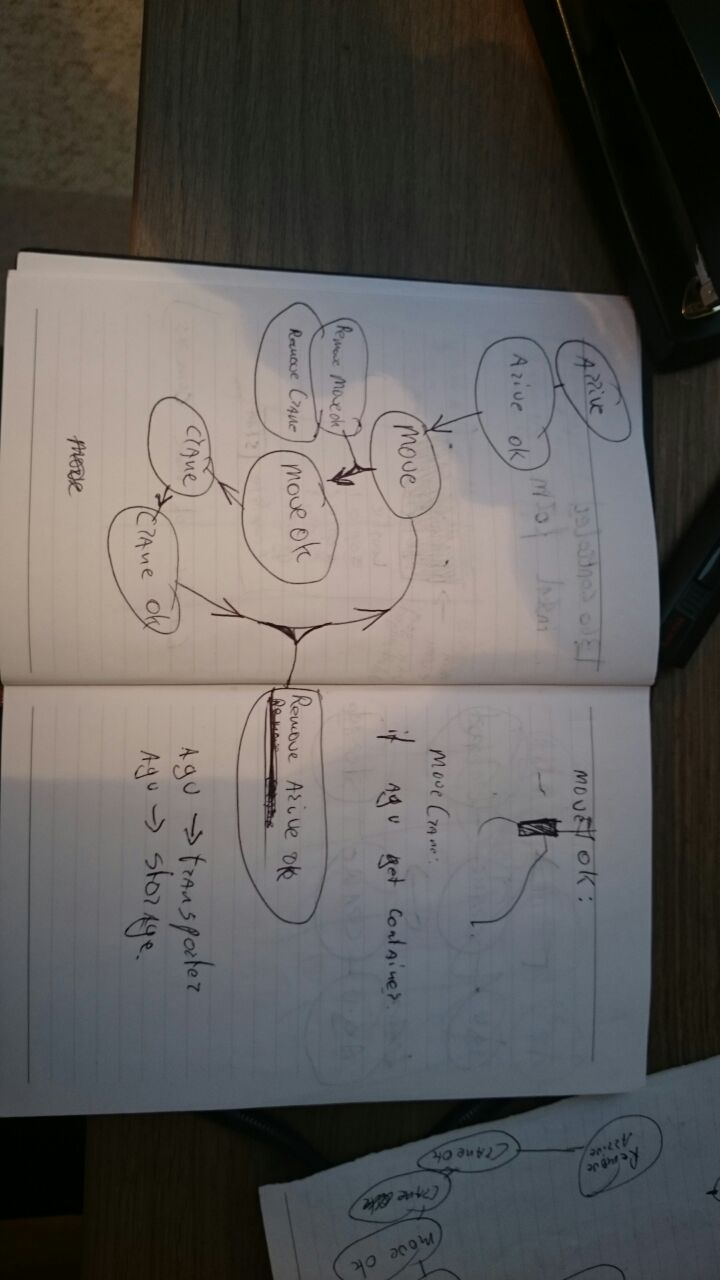


Figuur 4 Klassediagram Simulator



Figuur 5 STD Controller & Simulator



Figuur 7 Threading Communicatie

Figuur 6 Communicatie Message systeem