

# Technisch verslag wasmachine

Team 10

Kevin Damen 1660811

Remco Nijkamp 1657833

Jordan Ramirez 1652760

Jeroen Kok 1666479

Versienummer 1.0



# 1 Managementsamenvatting

In opdracht van de Hogeschool van Utrecht moest er een webapplicatie gemaakt worden die op afstand een wasmachine kan aansturen. Hierdoor kunnen mensen hun wasmachine van te voren vullen en dan via hun computer aan kan zetten. Hierdoor kunnen bijvoorbeeld de kinderen de was in de wasmachine stoppen en dan de moeder die op dat tijdstip aan het werk of misschien aan het koken is de keuken de wasmachine aanzetten. Wat natuurlijk heel erg handig kan zijn en veel tijd kan schelen.

Om dit product op te kunnen leveren, moest veel informatie verkregen worden. Zo moesten eisen worden gesteld met de klant over het functioneren van de wasmachine en de bijbehorende webapplicatie aan de hand van een interview.

Naar aanleiding van dit interview is een MoSCoW opgesteld en vond terugkoppeling plaats met de klant.

Vervolgens is de verkregen informatie uitgewerkt in een zogenaamde requirements architecture. Het belangrijkste dat hierin naar voren kwam is manier waarop de scheiding wordt getrokken tussen de webapplicatie en de wasmachine. Dit is het beste terug te vinden in de activity diagrams.

Aan de hand van de opgestelde requirements architecture is een solution architecture gemaakt. De belangrijkste ondervinding voor het systeem die hier naar voren kwam, is de taakstructuur. Door te werken met aparte taken voor de verschillende elementen in de wasmachine (temperatuur, motor, waterniveau) kan een controller worden ingezet die puur een leidinggevende functie heeft over deze taken.

Na het opstellen van de solution architecture kon het systeem daadwerkelijk gebouwd worden. Als snel bleek een zich een cyclische dependency te ontstaan binnen de code. Eerst werd gekeken hoe deze weg kon worden genomen, maar dit zou het belangrijkste idee over de taakstructuur weghalen. Daarom is uiteindelijk gekozen deze cyclische dependency toch te behouden en een manier te vinden dit zo netjes mogelijk op te nemen binnen de code.

De wasmachine bleek naar behoren te werken. Wel is er door uitval van verscheidene teamleden en hier en daar wat tegenslagen, te weinig tijd geweest om alle punten die we voor ogen hadden volledig af te ronden. De basis daarvoor is echter wel aanwezig en zeker zichtbaar en met wat meer tijd was dit zeker gelukt. Al met al functioneert het product wel zoals gehoopt was op deze kleine punten na.

## [1 Managementsamenvatting](#)

## [2 Inleiding](#)

## [3 Onderzoek](#)

## [4 Requirements architecture](#)

### [4.1 Use Case Diagram:](#)

#### [4.1.1 Keuzes structuur](#)

#### [4.1.2 Toelichting use case diagram](#)

##### [4.1.2.1 Wasprogramma draaien](#)

##### [4.1.2.2 Wasprogramma afbreken](#)

##### [4.1.2.3 Was log weergeven](#)

##### [4.1.2.4 Systeem updaten](#)

##### [4.1.2.5 Wasprogramma fout](#)

##### [4.1.2.6 Wasmachine besturing](#)

##### [4.1.2.7 Gebruiker](#)

##### [4.1.2.8 Fabrikant](#)

### [4.2 Activity Diagram:](#)

#### [4.2.1 keuzes Diagrammen](#)

##### [4.2.1.1 UserInterface](#)

##### [4.2.1.2 WasprogrammaDraaien](#)

##### [4.2.1.3 WasmachineBesturing](#)

### [4.3 Constrains Model](#)

#### [4.3.1 Performance](#)

#### [4.3.2 Accuracy](#)

#### [4.3.3 Resource Use](#)

#### [4.3.4 Availability](#)

#### [4.3.5 Reliability](#)

#### [4.3.6 Learnability](#)

## [5 Solution architecture](#)

### [5.1 Klassendiagram](#)

#### [5.1.1 Wasmachine deel](#)

#### [5.1.2 Webapplicatie deel](#)

### [5.2 Taakstructurering](#)

### [5.3 Concurrency Model](#)

#### [5.3.1 Communicatie middelen](#)

#### [5.3.2 Diagram](#)

### [5.4 Dynamic Model](#)

#### [5.4.1 Uart](#)

#### [5.4.2 WebController](#)

#### [5.4.3 WashingMachineController](#)

#### [5.4.4 MotorController](#)

#### [5.4.5 WaterController](#)

#### [5.4.6 TempController](#)

## [6 Webapplicatie](#)

### [6.1 Eisen](#)

### [6.2 Keuzes](#)

## [7 Realisatie](#)

### [7.1 Werkomgeving](#)

#### [7.1.1 De website](#)

#### [7.1.2 De modellen](#)

#### [7.1.3 De code ontwikkelomgeving](#)

#### [7.1.3 Raspberry PI](#)

### [7.2 Uitzonderlijke code](#)

## [8 Evaluatie](#)

### [8.1 Problemen & Oplossingen](#)

## [9 Conclusies en aanbevelingen](#)

## [10 Bronvermeldingen](#)

## [Bijlagen](#)

### [I MosCow](#)

### [II Use Case Diagram](#)

### [III Activity Diagrams](#)

#### [III.I User Interface](#)

#### [III.II Wasmachine Besturing](#)

#### [III.III Wasprogramma Draaien](#)

### [IV Constraints Model](#)

### [V Klassendiagram](#)

#### [V.I Wasmachine deel](#)

#### [V.II Webapplicatie deel](#)

### [VI Concurrency Diagram](#)

#### [VI.I Hele diagram](#)

### [VII State Transition Diagram](#)

#### [VII.I UART](#)

#### [VII.II WebController](#)

#### [VII.III WashingMachineController](#)

#### [VII.IV MotorController](#)

#### [VII.V WaterController](#)

#### [VII.VI TempController](#)

## 2 Inleiding

Het wassen van kleding, het wordt al decennia lang gedaan. Vele jaren werd het gewoon met de hand gedaan. Maar in de begin 20ste eeuw is er een uitvinding gedaan die dat allemaal een stuk makkelijker maakte: “de wasmachine”. Het doel is om deze alledaagse activiteit nog gemakkelijker te maken door middel van een “webapplicatie” waarmee je de was kan starten en bijhouden op afstand!

Elektronica is niet meer weg te denken in de huidige maatschappij. Het wordt allemaal steeds normaler. Zelfs het oude vrouwtje op de hoek van je straat heeft tegenwoordig een smartphone. Steeds meer dingen worden vernieuwd en kan je online, op de webbrowser of met de applicatie op je telefoon of tablet, aansturen. De wasmachine kan hierin natuurlijk niet achterblijven.

In opdracht van de Hogeschool Utrecht moet in blok 2 van het tweede leerjaar een project worden uitgevoerd omtrent een wasmachine. In dit project moet een webapplicatie worden gemaakt die op afstand een wasmachine moet kunnen aansturen.

Alles in het project is gedocumenteerd, waardoor er goede informatie gegeven kan worden wat er is gedaan tijdens deze projectweken.

Het doel van dit verslag is het informeren van de lezer, hoe het project is verlopen, hoe het project is gemaakt en welke stappen daarvoor zijn uitgevoerd. Deze stappen worden in hoofdstukken aangegeven. Hieronder volgt per hoofdstuk een korte omschrijving om de lezer een indruk te geven wat deze hier kan verwachten.

In hoofdstuk 3, Onderzoek, zal worden besproken welke vormen van onderzoek zijn gebruikt voor het verkrijgen van de nodige informatie. Zowel de informatie die nodig was om starten te gaan met het project, als de informatie die naar voren kwam door middel van experimenten en testen over de werking van het product. Voor veel informatie is een interview gehouden met de klant. Wat hierin is besproken zal ook in dit hoofdstuk worden voorgelegd.

In hoofdstuk 4, Requirements architecture, zal worden besproken welke eisen er zijn gesteld aan het programma en wat de gebruiker met van het programma verwacht. Door middel van (d.v.m) het Use Case Diagram wordt uitgelegd wat de gebruiker van het systeem wilt. Met het Constraints Diagram wordt uitgelegd aan welke eisen het systeem moet voldoen. D.v.m het Activity Diagram wordt duidelijk welke stappen het systeem doorloopt tijdens het gebruik.

In hoofdstuk 5, Solution architecture, wordt er gesproken over de bouw en werking van het daadwerkelijke programma zelf. Er wordt gesproken over de structuur van het programma d.v.m het klassendiagram, de samenwerkings technieken d.v.m de concurrency diagram en de verschillende states d.v.m het state transition diagram.

In hoofdstuk 6, Realisatie, zal er besproken worden wat er allemaal goed en fout is gegaan. Welke fouten zijn gemaakt, hoe deze zijn opgelost en welke nog niet zijn opgelost. Ook worden er stukken code besproken die in tweede of derde instantie niet duidelijk zijn en niet van zichzelf spreken. ook worden de tests die zijn gedaan besproken.

In hoofdstuk 7, Evaluatie, zal worden besproken welke dingen bij nader inzien beter hadden gekund of anders opgelost zijn en hoe deze dingen beter hadden gekund.

Tot slot zal in het laatste hoofdstuk, Conclusies en aanbevelingen, worden gekeken naar zaken die beter hadden kunnen gaan tijdens dit project. Dit zijn zaken zoals taken die bijvoorbeeld beter door een ander groepslid gedaan had kunnen worden, of zaken die beter gecommuniceerd hadden moeten worden.

Hopelijk bent u hiermee goed geïnformeerd over wat u kunt verwachten in dit verslag. heel veel lees plezier.

### 3 Onderzoek

Alvorens te werk te kunnen gaan aan de wasmachine en de webapplicatie, was het nodig om eerst met de opdrachtgever om tafel te gaan zitten en een aantal punten door te nemen. Dit is gebeurd aan de hand van een interview.

Tijdens dit interview zijn veel zaken aan bod gekomen, zo heeft de klant verteld wat hij verwacht van het eindproduct en hoe hij voor zich ziet dat de webapplicatie moet werken.

Om te beginnen is grondig besproken hoe de wasmachine zelf moet functioneren.

Hierin is naar voren gekomen dat dit wel vrij standaard is. Hij wordt in elkaar gezet met de standaard onderdelen, Maar verder kwam er weinig aan te pas.

Het team heeft na een kleine verdieping in stof over de wasmachine, de opdrachtgever ook advies gegeven over wat wel dan wel niet haalbaar zou kunnen zijn door middel van een MoSCoW. (bijlagen: 1 MoSCow)

Een model waarin wordt aangegeven aan de klant wat de klant wil van zijn product.

MoSCoW is een afkorting waar de letters M, S, C en W staan voor: de M staat voor Must. deze punten moeten worden opgeleverd. zonder die eisen werkt het product niet zoals het zou moeten.

dan heb je de S die staat voor Should. in dit rijtje heb je eisen die niet perse moeten maar de klant wel graag in zijn product wil zien. dan werkt het product dus al wel maar het is bijvoorbeeld nog niet mooi of snel genoeg. Dan komen we bij de C die staat voor Could. deze punten worden eigenlijk pas gedaan wanneer je helemaal klaar bent met de eisen van de Must en Should en er is dan nog tijd over tot de deadline. Als laatste letter is de W die voor Won't staat. Dit zijn punten waar je op het begin al zeker weet dat je er niet aan toe komt. dus ook niet worden gedaan. Hierdoor wordt de klant heel goed geïnformeerd wat hij nou na de deadline van zijn project kan verwachten.

Vervolgens is de webapplicatie besproken. Hoe deze zal moeten samenwerken met de wasmachine was een eerste punt van gesprek. Dit zou dus moeten door een connectie te maken met de raspberry pi. Dat was al begrepen. Maar wat moet de webapplicatie allemaal kunnen en hoe moet het eruit komen te zien. Ten eerste was het al duidelijk dat je via webapplicatie een wasprogramma kan invoeren en deze dan kan laten draaien. Dit mocht niet zomaar, want de opdrachtgever wilde dat het systeem wel beveiligd zou zijn met een inlog code. Wat de opdrachtgever ook heel duidelijk heeft gezegd is dat de webapplicatie overzichtelijk moet zijn. Iedereen zou gelijk door moeten hebben hoe de webapplicatie werkt. Dit betekende dat alles op het scherm moet passen en dat er dus geen scrollen bij te pas zou komen.

Tot slot is besproken wat een gebruiker allemaal moet kunnen doen vanaf de webapplicatie. En ook wat moet vooral niet kunnen. De klant zou graag een logging willen zien op de webapplicatie. Dat betekend dat je kon zien hoeveel water en energie er verbruikt zou zijn. Wat de gebruiker juist niet mocht doen: Is alle gewenste temperaturen, water hoeveelheden

en duur kiezen. De gebruiker moest gelimiteerd worden tot een aantal opties. Hierdoor zou er minder fout lopen tijdens het wassen van de kleding.

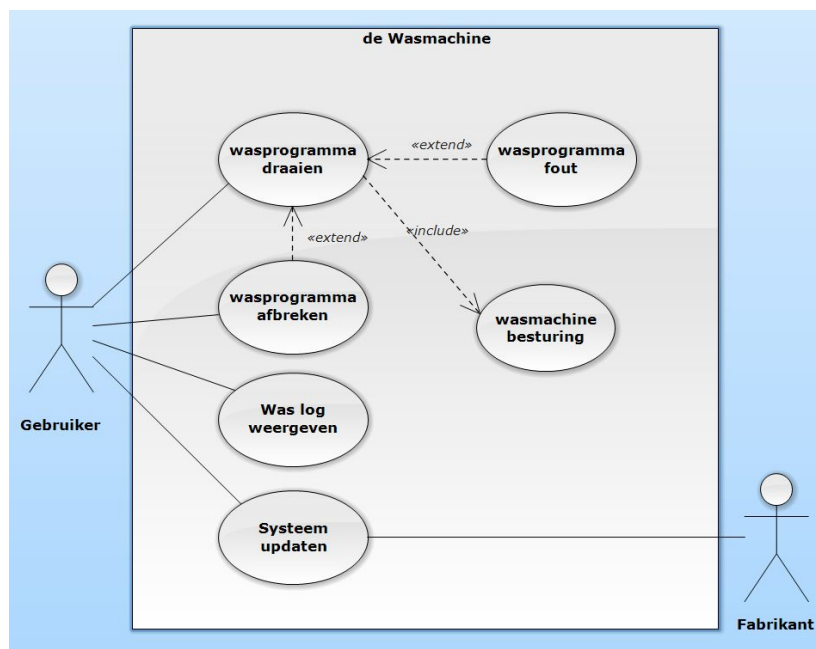
## 4 Requirements architecture

Het doel van de requirement architecture (RA) is het vastleggen van de functionele systeemeisen. wat is de bedoeling dat de software van het systeem doet? In het RA wordt ook beschreven hoe goed het systeem moet functioneren. De requirement architecture is een belangrijk middel voor het overleg met de klant over het te maken systeem die bestaat uit het use case diagram, activity diagram en het constraints model.

### 4.1 Use Case Diagram:

De use case diagram legt vast wat het systeem moet doen door middel van use cases en actoren. Bij het maken van de use cases voor het systeem is in feite het aanbrengen van de structuur in de verschillende acties die het systeem moet doen. Hierbij worden de acties die bij elkaar horen aan elkaar gelinkt. een use case is:

*“Een samenhangende reeks van acties die door het systeem worden uitgevoerd om aan een bepaald doel van een actor te voldoen.”*



#### 4.1.1 Keuzes structuur

Bij het maken van het use case diagram zijn keuzes gemaakt die invloed hadden op het uiteindelijke uiterlijk van het diagram. Deze keuzes worden eerst besproken.



Zoals te zien in het use case diagram (*bijlagen: 1.1 Use Case Diagram*) zijn er twee actoren opgenomen. Deze actoren zijn de gebruiker en de fabrikant. De gebruiker neemt hierin de rol van primaire actor op zich en de fabrikant de rol van secundaire actor. Hiermee wordt bedoeld dat de gebruiker het initiatief tot de acties neemt. Het kan nodig zijn dat een monteur de wasmachine moet besturen als de wasmachine mankementen vertoont. Voor de rol van de monteur is geen extra actor opgenomen in het diagram, aangezien de monteur in principe dezelfde acties kan uitvoeren als de gebruiker. Het verschil tussen de gebruiker en de monteur is dat bij het tonen van de logs, de monteur andere logs te zien krijgt dan de gebruiker. Welke logs er getoond moeten worden, weet het systeem af te leiden van welke gebruiker is ingelogd. Hiermee is de monteur dus niet meer dan een speciale vorm van een gebruiker.

In dit use case diagram gaan we ervan uit dat bij gebruik van de van de webapplicatie de gebruiker eerst is ingelogd alvorens hij/zij hier gebruik van kan maken. In de activity diagrammen is het inloggen als actie wel opgenomen, omdat hier de flow die de gebruiker als het ware veroorzaakt uiteen wordt gezet. Om het beeld compleet te maken is het inloggen en uitloggen hierin opgenomen als onderdeel van wasprogramma draaien.

## 4.1.2 Toelichting use case diagram

Onder dit kopje wordt de use case diagram helemaal beschreven wat het doel is van elke use case en actoren.

### 4.1.2.1 Wasprogramma draaien

Het doel van deze use case is het draaien van een wasprogramma op de wasmachine die ingesteld wordt via de webapplicatie.

### 4.1.2.2 Wasprogramma afbreken

Wanneer er op de noodstop-knop wordt geklikt dan wordt moet het wasprogramma direct worden afgebroken. Hierbij mag de wasmachine niet meer centrifugeren, er mag geen water meer in de wasmachine zitten en het verwarmingselement moet uit zijn. Daarnaast als dat is gebeurt moet de deurvergrendeling van het slot, zodat de deur open gemaakt kan worden.

### 4.1.2.3 Was log weergeven

Het weergeven van de was gegevens is een doel die gedaan kan worden op de webapplicatie. Deze logs gegevens worden opgeslagen in een apart bestand op de raspberry pi. Dan op zijn beurt die gegevens dan weer doorstuurt naar de webapplicatie.

### 4.1.2.4 Systeem updaten

Het doel van deze use case is het updaten van het systeem, wanneer de webapplicatie is aangepast dan moet de gebruiker de keuze moeten hebben om dit te kunnen weigeren. Om niet de nieuwe wasprogramma's te willen hebben.

#### 4.1.2.5 Wasprogramma fout

Als er een fout is met de hardware van de wasmachine of er is een software fout die het systeem zelf niet kan oplossen dan wordt deze use case gebruikt. Dat betekent dat er geen wassen mogen gedraaid mogen worden.

#### 4.1.2.6 Wasmachine besturing

In deze use case is een opvolger van de andere use case “Wasprogramma draaien”. Het doel van de use case “Wasmachine besturing” is was wat de naam al zegt het aansturen van de wasmachine. Hieronder bedoelen we het centrifugeren, het verwarmen, het toevoegen van het water en zeep en het vergrendelen van de deur.

#### 4.1.2.7 Gebruiker

de actor “Gebruiker” is de gebruiker van de webapplicatie. De gebruiker stelt dus alles in op webapplicatie en start op deze manier de wasmachine.

#### 4.1.2.8 Fabrikant

De Fabrikant zijn de ontwerpers van de wasmachine en de webapplicatie. Die voor nieuwe software kan zorgen voor het product.

## 4.2 Activity Diagram:

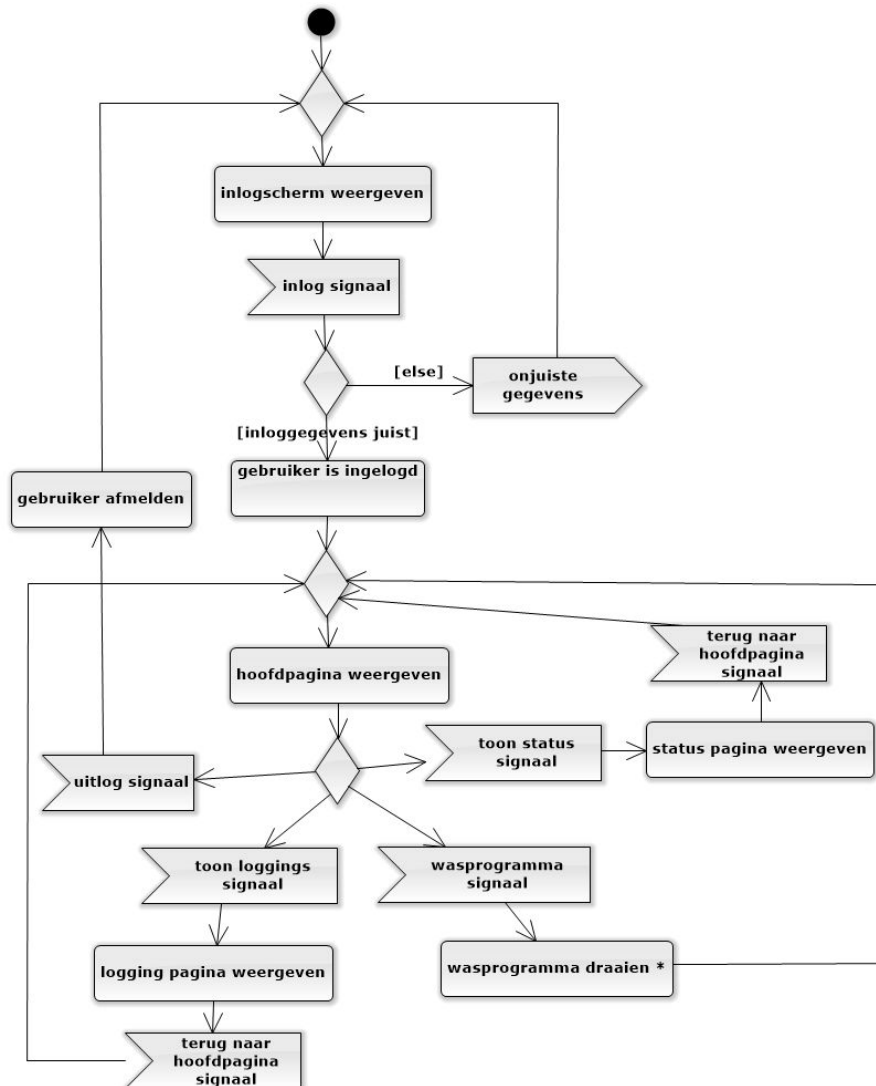
Het activity diagram is het volgende model in de requirement architecture. Dit is het gevolg van de use case diagram. Een use case bestaat uit een reeks van acties maar aan de use case diagram zie je dat niet echt duidelijk, daarom is de activity diagram die laat doormiddel van nodes en flows te laten zien hoe die acties van de use case worden uitgevoerd en wanneer ze worden uitgevoerd.

### 4.2.1 keuzes Diagrammen

Voor de uitwerking van de activity diagrammen, is gekozen voor het maken van drie diagrammen. Met het oog op de use case diagram die tot stand is gekomen, is dit niet de meest voor de hand liggende oplossing, maar er zal worden uitgelegd waarom toch voor deze aanpak is gekozen.

De drie diagrammen die tot stand zijn gekomen zijn UserInterface, WasmachineBesturing en WasprogrammaDraaien.

#### 4.2.1.1 UserInterface



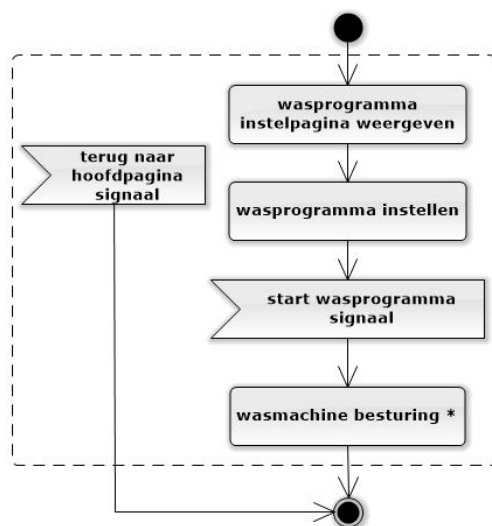
In dit diagram (*bijlagen: III.1 User Interface*) wordt als het ware gestart met lezen. Wat opvalt is dat hij wel een start node heeft, maar geen end node. Er wordt een cyclus doorlopen. Als wordt begonnen bij de start node, wordt meteen het inlogschermbalk weergegeven. Als de gebruiker vervolgens inlogt, komt er een inlog signaal. Na het ontvangen van dit signaal wordt gecheckt of de inloggegevens juist waren. Was dit het geval, dat is de gebruiker ingelogd, zo niet, dan wordt opnieuw het inlogschermbalk getoond en kan de gebruiker het opnieuw proberen.

Nadat de gebruiker is ingelogd, wordt de hoofdpagina getoond. Vanuit hier kan de gebruiker een aantal dingen doen. Hij kan kiezen om weer uit te loggen, waarna een uitlog signaal wordt afgevangen. De gebruiker wordt dan weer afgemeld en de hoofdpagina zal weer getoond worden.

De gebruiker kan na inloggen ook kiezen dat hij de loggings wil inzien. Nu zal er een signaal toon loggings worden gestuurd, waarna de logging pagina zal worden weergegeven. Als de gebruiker vervolgens een signaal geeft dat hij terug wil naar de hoofdpagina, zal de hoofdpagina opnieuw worden getoond.

Een andere keuze die de gebruiker heeft na inloggen is het zien van de status van de wasmachine. Na het signaal voor het tonen van de status, zal de statuspagina worden weergegeven. Vervolgens kan de gebruiker aangeven dat hij terug wil naar de hoofdpagina, waarna deze, na het ontvangen van het signaal hiervoor, zal worden getoond. De laatste, en meest belangrijke optie die de gebruiker heeft is het aanzetten van een wasprogramma. In het diagram staat hier een sterretje achter. Hiermee wordt bedoeld dat wordt overgegaan naar het activity diagram WasProgramma Draaien. Op het moment dat dit diagram zijn end node heeft bereikt, zal worden teruggekeerd naar de hoofdpagina.

#### 4.2.1.2 WasprogrammaDraaien



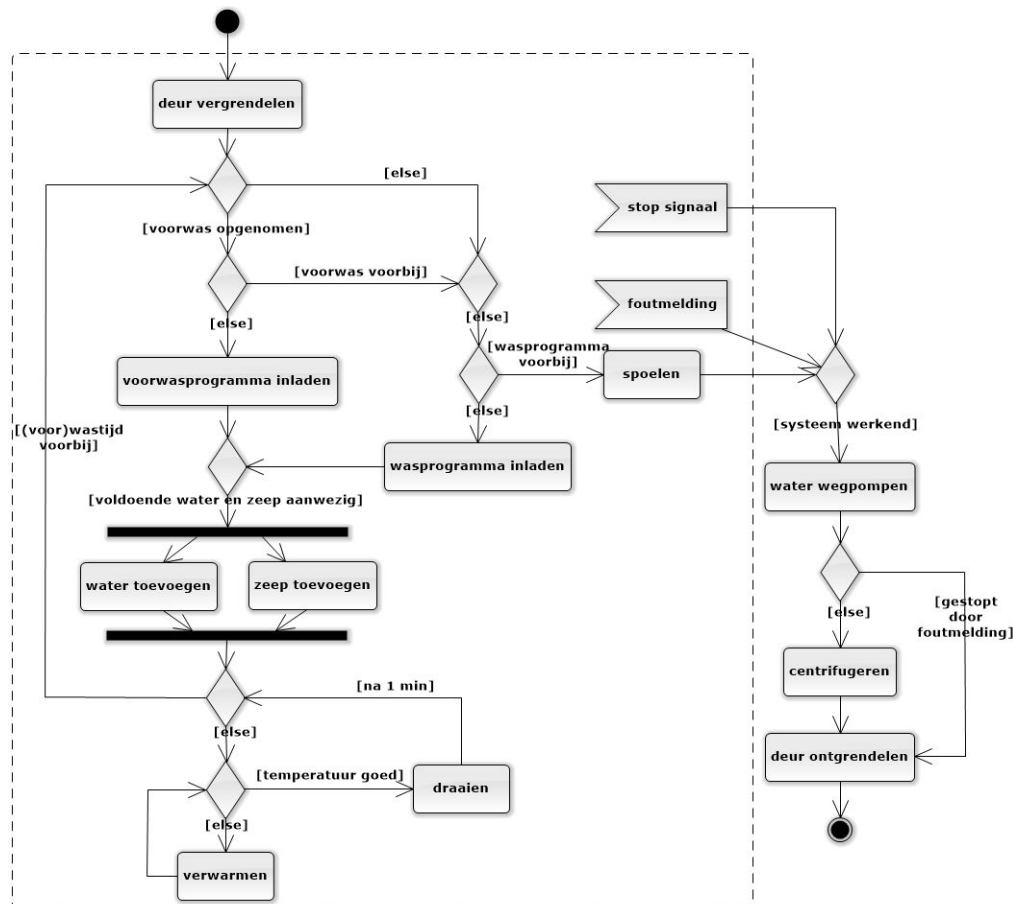
Nadat in de activity diagram *UserInterface* de *WasprogrammaDraaien* is bereikt, zal deze activity diagram beginnen vanuit zijn start node.

Dit deel (*bijlagen: III.II Wasmachine Besturing*) gaat nogsteeds uit van de webapplicatie.

Meteen zal de instelpagina voor het wasprogramma worden weergegeven. De gebruiker kan nu een wasprogramma instellen. Nadat de gebruiker klaar is met het instellen hiervan, geeft hij aan dat hij het wasprogramma wil starten. Een signaal zal worden afgevangen die aangeeft dat het programma gestart mag worden. Er zal nu worden verder gegaan met de activity diagram *WasmachineBesturing*. Nadat in deze activity diagram de end node wordt bereikt, zal ook hier worden overgegaan naar de end node en uiteindelijk terug worden gekeerd naar de activity diagram *UserInterface*.

Ten alle tijden van het instellen van het wasprogramma, zal de gebruiker kunnen besluiten dat hij toch liever terug wil keren naar de hoofdpagina. In dit geval zal een signaal worden afgevangen die eindigt in de end node van deze activity diagram.

#### 4.2.1.3 WasmachineBesturing



Nadat in de activity diagram WasprogrammaDraaien de WasmachineBesturing is bereikt, zal deze activity diagram (*bijlagen: III.III Wasprogramma Draaien*) beginnen vanuit zijn start node.

Meteen zal de deur worden vergrendeld. Er wordt dus vanuit gegaan dat de was al in de wasmachine is gestopt, alvorens het programma te starten. Anders zal de wasmachine zonder was gaan draaien, bijvoorbeeld om zichzelf te reinigen.

Nadat de deur is vergrendeld zal worden gecheckt of een voorwas is ingesteld. Als dit het geval is, zal deze worden uitgevoerd, is dit niet het geval, dan zal meteen worden overgegaan naar het hoofdwasp programma.

In principe wordt voor het voorwasprogramma en het hoofdwasp programma dezelfde cyclus doorlopen. Het verschil hierbij is dat voor beide een ander voorschrift bestaat. Zo zal bij beiden water worden toegevoegd, maar bij de een zal dit meer zijn dan bij de ander. Ook kan bijvoorbeeld de temperatuur verschillen of de tijd dat hij moet draaien.

Stel dat er een voorwasprogramma aanwezig was, dan zal deze eerst worden ingeladen. Vervolgens wordt gekeken of de watertoevoer beschikbaar is en of er voldoende zeep aanwezig is om dit programma te kunnen draaien. Als dit het geval is, worden water en zeep toegevoegd.

Vervolgens zal het water worden verwarmd tot de gewenste temperatuur. Nu zal de wasmachine gaan draaien. Elke minuut zal even een check plaatsvinden of de temperatuur

nog op peil is. Is dit het geval dan draait de wasmachine gewoon door. Is dit niet het geval, dan wordt het water verwarmd.

Nadat het programma voorbij is zal opnieuw worden gekeken of een voorwas is opgenomen en zo ja of deze al voorbij is. Vervolgens wordt dezelfde cyclus doorlopen voor het hoofdwasp programma. Als ook deze voorbij is zal worden gespoeld. Hierna wordt het water weggepompt en gecentrifugeerd, waarna de was klaar is en de deur zal worden ontgrendeld.

De end node is nu bereikt en er zal worden teruggekeerd naar de activity diagram WasprogrammaDraaien.

Tijdens de gehele wasprogramma cyclus (dus inclusief voorwasprogramma), dan er een fout optreden of op de noodknop (stop) worden gedrukt. In dit geval wordt de activiteit stilgezet en zal worden gekeken of het systeem nog kan functioneren. Is dit het geval dan zal het water worden weggepompt en de deur ontgrendeld. Functioneert het systeem daarentegen niet, zal net zolang worden gewacht tot het systeem weer werkt.

## 4.3 Constrains Model

De requirements architecture dient alle functionele eisen die door de gebruiker aan het systeem worden gesteld vast te leggen. In het constraints model (*bijlagen: IV Constraints Model*) staan juist de niet-functionele eisen die aan het systeem worden gesteld. Dit zijn waarneembare eigenschappen van het systeem. Deze eigenschappen zijn vaak meetbaar. Hiermee bedoelen we bijvoorbeeld timing: hoe snel moet een systeem reageren? of zuinigheid: hoe veel stroom verbruikt het systeem? Dit wordt dan in een overzichtelijke tabel gezet. Waardoor het duidelijk is welke niet-functionele eisen aan het systeem zijn gebonden.

### 4.3.1 Performance

Performance houdt de snelheid van iets in - hoeveel tijd deze mogen kosten; leessnelheid, schrijfsnelheid, reactietijd.

Tussen de communicatie tussen de server en de webapplicatie bestaat een vertraging vanwege het internet. Deze vertraging kan storend zijn voor de gebruiker als deze de webapplicatie wilt navigeren maar is minder erg als de was 1 minuut later start dan dat er op de knop is ingedrukt. Om dit te verhullen is er latency-hiding. Als de gebruiker op de knop drukt dan lijkt het alsof er iets gebeurt maar in werkelijkheid is de server nog bezig de request te verwerken. Dingen zoals het (direct) starten van de was kan hiermee gedaan worden. Als de gebruiker op de startknop drukt wilt deze direct feedback hebben dat hij op de knop heeft gedrukt. Maar dat de was 20 tot 30 seconden later start valt veel minder op. Deze constraint is daarom gesplitst in 2 delen; de reactietijd van de server en de reactietijd van de webapplicatie. De eerste betekent: hoelang duurt het voordat de wasmachine reageert op acties vanuit de webapplicatie. De tweede is de reactietijd van de webapplicatie op een actie vanuit de webapplicatie. Als een knop is ingedrukt om te starten, een laadpagina weergeven bijvoorbeeld.

Deze constraint is simpelweg te meten door middel van een stopwatch. "Reactietijd webapplicatie" is wat lastiger door de constraint onder de seconden, maar zou toch prima te doen moeten zijn.

### 4.3.2 Accuracy

Accuracy is de nauwkeurigheid waarmee het systeem werkt. Bijvoorbeeld hoeveel decimalen achter de punt de temperatuur wordt weergegeven en of er secondes bij een countdown staan of alleen de minuten.

Zowel op de webserver en op de browser waar de webapplicatie op wordt afgebeeld staan klokken. Een zo een klok is de countdown totdat de was klaar is. In een perfecte wereld zouden deze twee altijd gelijk lopen als je op hetzelfde moment zegt dat de countdown moet starten. Helaas is dit niet het geval en zullen de klokken een kleine afwijking opbouwen ten opzichte van elkaar. Deze afwijking kan bijvoorbeeld 10 seconden per uur zijn of meer.



Een tragere browser zal moeite hebben met de tijd exact af te tellen. Daarom is besloten dat deze afwijking niet meer dan 1 minuut per uur mag afwijken. Hiervoor is gekozen omdat een webpagina in de regel niet meer dan een uur aan zal staan en een minuut of 2 niet zo opvalt. als 5 of 7 minuten.

Of aan deze constraint is voldaan wordt de tijd een uur lang weergegeven en dan gekeken hoeveel seconden de tijd is afgeweken. Mocht deze te veel afwijken dan kunnen er verschillende synchronisatie technieken worden toegepast.

### 4.3.3 Resource Use

Resource Use is de hoeveelheid er van iets gebruikt wordt. Zoals watergebruik, energieverbruik en RAM gebruik.

Energiegebruik is op de dag van vandaag een gevoelig onderwerp. De opwekking van elektriciteit levert een groot deel van de broeikasgassen op die voor de verwarming van de aarde zorgen [bron nodig]. En hoewel er een veel grotere hoeveelheid energie beschikbaar zou zijn als er niet allerlei 'groene' regels waren, moet er rekening houden moet hoeveel energie er wordt gebruikt. Daarnaast is een lagere elektriciteitsrekening ook wel fijn. Daarom is besloten dat de wasmachine niet meer dan 1 kWh zal gebruiken. Dit is een goede balans tussen een reëel verbruik en een niet verspillend verbruik.

Het energieverbruik kan simpelweg getest worden door energiemeters aan te sluiten tussen de wasmachine en het lichtnet.

### 4.3.4 Availability

Availability is de hoeveelheid tijd die een systeem beschikbaar is in een bepaalde periode. Bijvoorbeeld een straatlantaarn die 99% van het jaar nachts aan staat.

De webapplicatie is de enige manier voor een gewone gebruiker om de wasmachine aan te sturen. Daarom is het prettig als de webapplicatie in normale omstandigheden altijd beschikbaar is. Er is besloten dat de webapplicatie altijd beschikbaar moet zijn als de wasmachine toegang heeft tot elektriciteit en netwerk. Mocht een van beide ontbreken dan kan er echter niet gegarandeerd worden dat het systeem werkt. Met een uitzondering voor de netwerktoegang. Mocht de wasmachine niet beschikbaar zijn dan moet de wasmachine toch zijn was af kunnen maken zonder problemen.

De beschikbaarheid van de webapplicatie is lastig te testen. Omdat dit over een langere periode gaat. De afhankelijkheid van de wasmachine aan het internet kan simpel worden getest door de netwerk kabel los te halen.

### 4.3.5 Reliability

Reliability is de betrouwbaarheid van een systeem. Bijvoorbeeld de stevigheid van autobanden zodat er geen klapband ontstaat.

Het is belangrijk dat de wasmachine robust is. Wasmachines zijn relatief duur en lastig te vervangen. Elk apparaat ondervindt echter slijtage en daar is weinig tegen te doen dan duurdere, sterkere materialen te gebruiken. Mocht er iets in de wasmachine kapot gaan dan is het de bedoeling dat de was onmiddellijk maar veilig wordt gestopt zodat het kapotte onderdeel niet meer schade kan aanrichten.

Met een fysieke wasmachine is dit lastiger te testen, maar met de emulator kunnen simpelweg pinnen worden verwijderd of het systeem daar goed op reageert.

### 4.3.6 Learnability

Learnability is hoe snel de gebruikers kan leren omgaan met een systeem. Bijvoorbeeld is een IKEA kast snel te maken, maar moet men meer tijd steken in het leren van Quantum Mechanica.

“Een kind kan de was doen” en dat is zeker waar met de webapplicatie. Het is belangrijk dat de gebruiker niet teveel tijd kwijt is aan het leren hoe de wasmachine werkt en deze zo snel mogelijk in gebruik kan nemen. Daarom is het belangrijk dat iedereen ook met de webapplicatie kan omgaan zonder de handleiding te hoeven te lezen. Uitzondering is wel dat de gebruiker met een pc/smartphone/tablet kan omgaan en simpel arbeid kan leveren: de was in de wastrommel stoppen en de was eruit halen.

Dit is te testen door iemand die zo min mogelijk weet van de wasmachine de was te laten doen.

## 5 Solution architecture

In een solution architecture draait het vooral om de vraag hoe iets in elkaar zit. Hoe de software van een systeem in elkaar moet zitten zodat aan de eisen, die in de requirements architecture naar voren zijn gekomen, kan worden voldaan.

Onder de solution architecture vallen drie modellen die leiden tot deze oplossing en uiteindelijk zullen resulteren in een aardig opzet voor de te schrijven code. Het eerste model is het klassendiagram, waarin alle objecten worden langsgegaan die in het systeem nodig zijn en worden ondergebracht in verschillende klassen. Het volgende model binnen de solution architecture is het concurrency model. Hierin wordt gekeken naar de verschillende taken die nodig zijn om het systeem te laten functioneren en naar de manier waarop deze taken met elkaar communiceren. Tot slot is er het dynamic model waarin het interne gedrag van de controller objecten wordt vastgelegd aan de hand van een aantal toestanden.

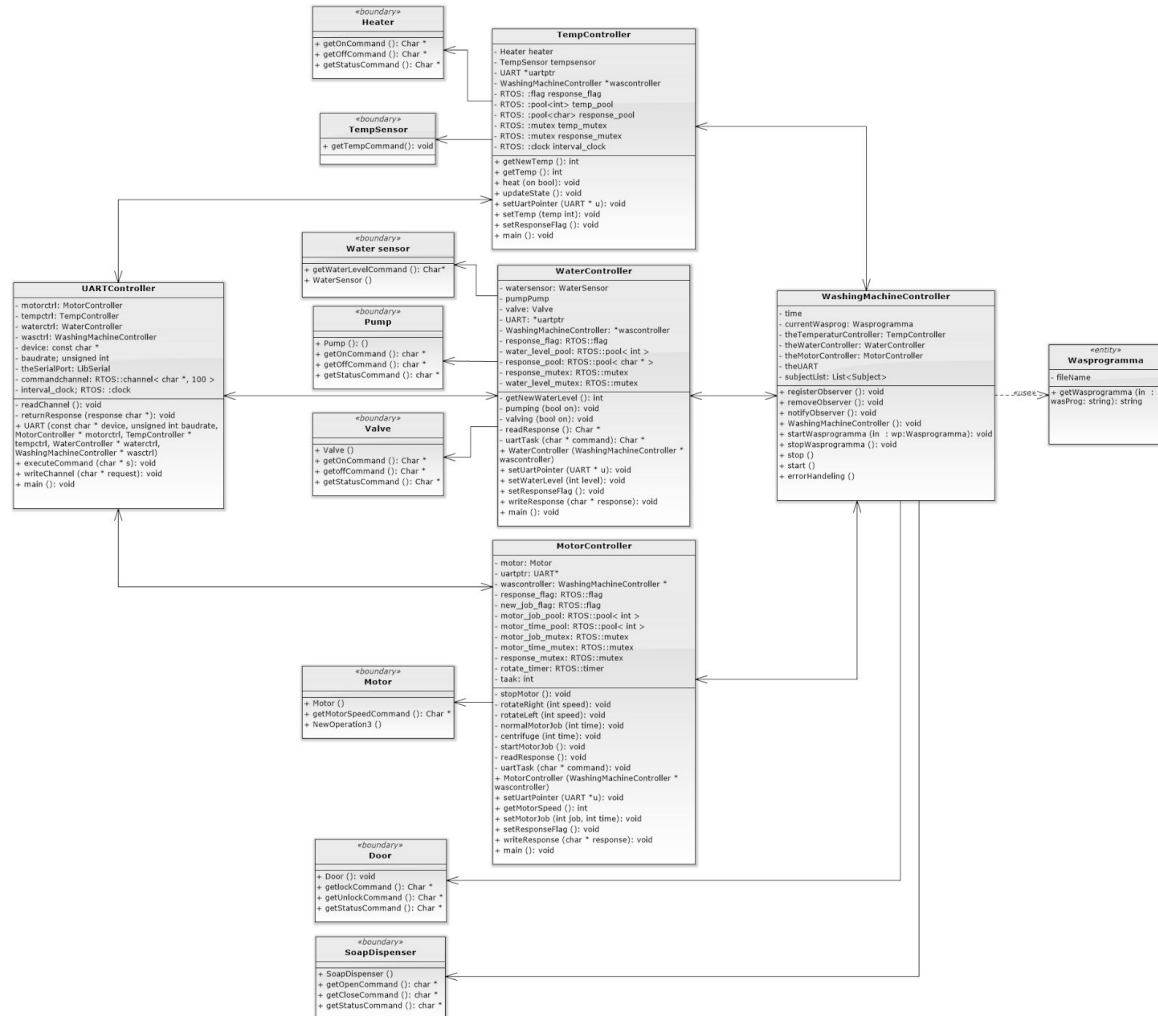
### 5.1 Klassendiagram

Zoals te zien in het klassendiagram (*bijlagen: V.III Hele diagram*) , bestaat het diagram uit twee delen: een deel voor de werking van de wasmachine, en een deel voor de communicatie met de webapplicatie.

Hieronder voor beide kanten een uitleg volgen van hoe het systeem in elkaar zit, hoe het globaal werkt, en wat de gedachten achter de gemaakte keuzes zijn.

### 5.1.1 Wasmachine deel

In het wasmachine deel (*bijlagen: V.I Wasprogramma deel*) bevinden zich de vijf controllers die de wasmachine aansturen. Dit zijn de WashingMachineController, de TempController, de WaterController en de MotorController. Daarnaast bevindt de Uart zich ook hier.



Voor de verwezenlijking van het wasmachine deel is gekozen voor één controller die het geheel stuurt, de WashingMachineController, met daaronder drie controllers die zorgen voor de besturing van de afzonderlijke boundary objecten.

Er is een TempController die alle zaken rond de temperatuur regelt. Deze controller klasse heeft twee boundary klassen onder zich, de Heater en de TempSensor.

De WaterController regelt op zijn beurt alle zaken die met het waterniveau te maken hebben. Onder hem vallen de boundary klassen WaterSensor, Pump en Valve.

Tot slot regelt de MotorController alle zaken die voor de motor zijn. Hij heeft dan ook de Motor als boundary klasse onder zich.

De Uart klasse verzorgt de communicatie over de seriële port. De Uart kent de drie controllers die samenwerken met de boundary klassen en er vindt wederzijdse communicatie plaats. Dus ook de controllers kennen de Uart.

De WashingMachineController staat in verbinding met de WebController.

De WebController vormt als het ware de scheiding tussen het wasmachine deel en het deel voor de webapplicatie. De WebController leest berichten die van de webapplicatie komen, met daarin informatie over het te draaien wasprogramma en commando's die uitgevoerd moeten worden, zoals het starten van een wasprogramma.

De WashingMachineController krijgt van deze WebController een nieuw wasprogramma om uit te voeren. Op zijn beurt coördineert de WashingMachineController dit wasprogramma door de onderliggende controllers verschillende taken te geven. Zo zal hij tegen de TempController zeggen dat een bepaalde temperatuur moet worden aangenomen. Dit geeft hij nu dus uit handen en de TempController zal verder regelen dat deze temperatuur bereikt zal worden en belangrijker nog, zal worden behouden.

De belangrijkste reden voor het maken van de drie controllers onder de WashingMachineController, was dat op die manier de temperatuur, het water niveau en de motortaken apart geregeld konden worden.

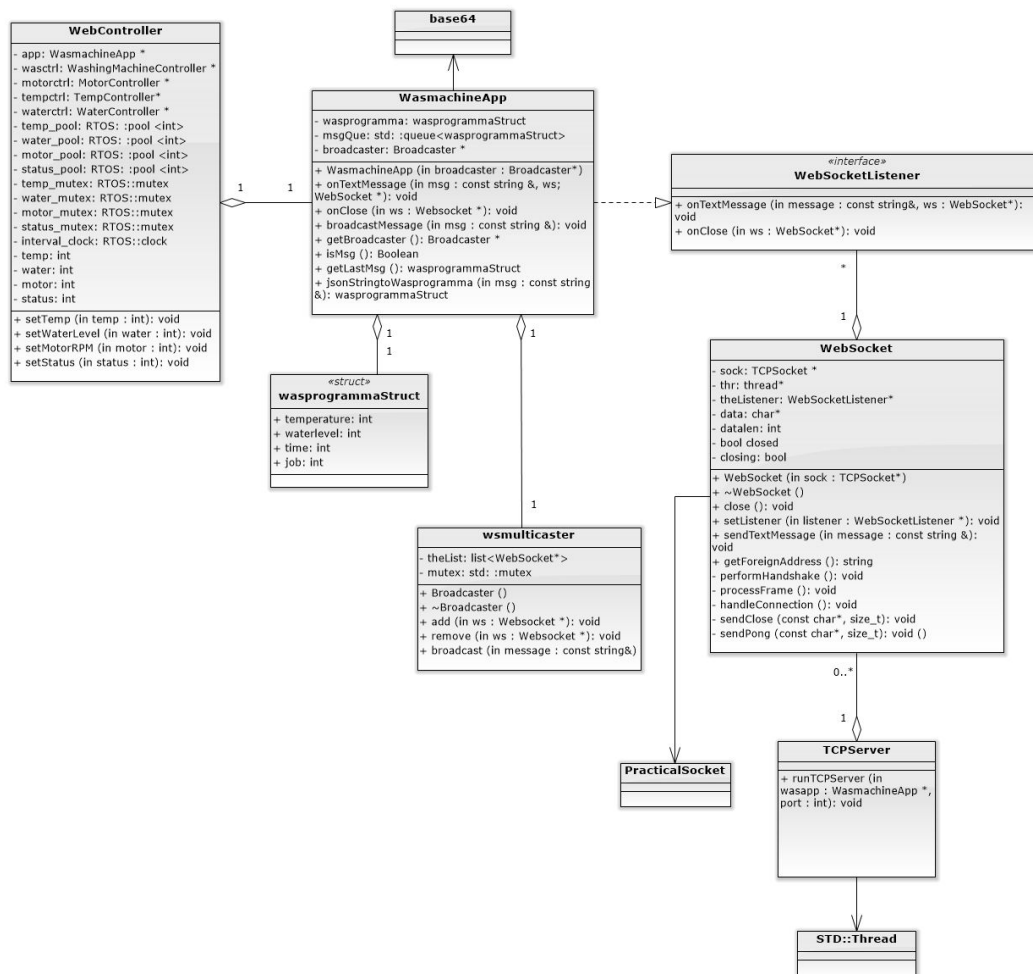
In het oorspronkelijke idee verliep de communicatie met de Uart via de boundary objecten. In dit geval zou de Uart niet de drie controllers kennen, maar alle boundary objecten. Echter bleek communicatie tussen de boundary objecten en de Uart niet heel makkelijk op deze manier. De Uart is een taak en kan niet zomaar worden onderbroken. Het idee was gebruik te maken van een channel en een aantal pools en mutexen om gegevens uit te kunnen wisselen, maar deze vormen van communicatie bleken enkel mogelijk te zijn tussen taken. Niet tussen een taak en een gewoon object. Daarom is er uiteindelijk voor de gekozen de verschillende controllers het commando dat richting de Uart gestuurd moet worden, op te vragen aan het boundary object van dien en deze vervolgens zelf richting de Uart te sturen.

De drie controllers kennen ook de WashingMachineController, aangezien ze signalen terug moeten kunnen geven, bijvoorbeeld of een temperatuur bereikt.

## 5.1.2 Webapplicatie deel

De wasmachine moest communiceren met het internet, daarvoor moest er een websocket beschikbaar zijn die dit kon regelen. Deze moest berichten kunnen ontvangen en versturen naar de webapplicatie(website). Vanaf de website wordt een string aan data verstuurd naar de websocket, de websocket moet deze inlezen en omzetten naar een stukje data waar de wasmachine wat mee kan. Vervolgens moet deze data opgevraagd kunnen worden vanaf de wasmachine kant. Omdat de websocket gebruik maakt van system calls van Linux kan dit niet in het RTOS draaien. Dit moet in een eigen thread van Linux zelf. Dit houdt ook in dat de websocket niets kan aanroepen bij alle klassen die in het RTOS systeem zitten. Wel kan het RTOS systeem bij de websocket het bericht ophalen en kijken of er een bericht is.

Als basis voor het systeem is de code gebruikt die dit blok bij het vak netwerk programmeren is meegegeven door Jan Zuurbier. Deze code is dit blok al gebruikt en de werking hiervan was daardoor al redelijk bekend. Deze code is voor een groot deel herschreven, waardoor het beter bruikbaar was binnen dit project.



## **WebController**

Dit is een RTOS task die verantwoordelijk is voor het pollen van data wat de website stuurt via de WasmachineApp classen. Deze task is nodig omdat een taak die buiten het RTOS staat niet aanroepen kan doen binnen een RTOS task. Dit zorgt voor problemen, daarom zorgt deze taak ervoor dat er periodiek gekeken wordt of er een nieuw bericht is in de WasmachineApp classen. Deze wordt dan opgehaald en doorgestuurd naar de juiste pools voor de wasmachine.

## **TCPServer**

Dit is de server die de verbindingen regelt naar de juiste sockets. Er wordt gekeken of er een nieuwe verbinding aangevraagd wordt op de juiste poort. Als dit het geval is zal er een websocket worden gemaakt. Deze klassen loopt in een thread omdat er constant gecontroleerd moet worden of er een nieuwe verbinding is gemaakt. Dit moet een Linux thread zijn omdat er ook gebruik gemaakt wordt van system calls naar Linux, als dit ook een RTOS task was geweest zou deze vast gelopen zijn.

## **WebSocket**

De websocket verzorgt de verbinding tussen de websocket en ons programma. Deze klasse wordt aangemaakt door de TCPServer en krijgt een TCPSocket binnen waarover de connectie moet plaatsvinden. Voor elke socket wordt een aparte thread gemaakt zodat alle verbindingen tegelijk kunnen werken. Als de websocket aangemaakt is, wordt de WebSocketListener eraan toegevoegd.

## **WasmachineApp**

Als er een bericht ontvangen wordt op de websocket, roept de WebSocket van zijn listener een functie aan. Aangezien de WasmachineApp een listener is van de websocket, wordt de bijbehorende code uitgevoerd. Deze code ontvangt het bericht en maakt hierin de vertaalslag van wat de website stuurt naar welke signalen de Wasmachine begrijpt. Als dit bericht vertaald is, wordt deze in een que gezet. Ook kan aan deze klasse gevraagd worden of er iets in de que staat. Als dit niet gebeurt kan het gebeuren dat als de que uitgelezen wordt er allemaal nullen of onzin gelezen wordt. Ook kan er een bericht worden verzonden naar alle aangesloten websockets.

## **Wsmulticaster**

Deze houdt de lijst bij van websockets waarnaartoe een bericht moet worden verzonden als er een broadcast wordt gedaan.

## **WebSocketListener**

Voor het listener pattern dat nodig is als er een bericht binnen komt moet er een juiste interface zijn zodat hetgeen wat het bericht ontvangt correct de listener aan kan roepen.

## **WasprogrammaStruct**

Van de webpagina komt een stukje data wat omgezet moet worden in een wasprogramma. Deze data wordt opgeslagen in deze struct.

## 5.2 Taakstructurering

Voor de realisatie van het systeem zijn een aantal taken in het leven geroepen. Elke taak is opgehangen aan een controller klasse.

De volgende taken zijn naar voren gekomen bij het uitdenken van het systeem, met daarachter welke klasse de rol van deze taak vervult en welke klassen ook onder deze taak vallen.

Taak voor de:

- UART (UART is de taak)
- wasmachine (WashingMachineController is de taak, Door en SoapDispenser vallen hier onder)
- motor (MotorController is de taak; klasse Motor valt hier onder)
- temperatuur (TempController is de taak; TempSensor en Heater vallen hier onder)
- waterniveau (WaterController is de taak; WaterSensor, Pump en Valve vallen hier onder)
- webapplicatie (WebController is de taak)

Er is dus voor gekozen om voor het aansturen van de wasmachine zelf, aparte taken voor de motor, de temperatuurregeling en de regeling van het water niveau. Hierboven bevindt zich één taak die zorgt voor de werking van de wasmachine in het algemeen.

De reden dat voor drie aparte kleinere taken is gekomen, is omdat zo verschillende kleinere taken met verschillende delen van de hardware bezig konden zijn. Zo zal de wasmachine taak de algemene sturing regelen en bijvoorbeeld op een bepaald moment de taak voor de temperatuur de opdracht geven dat een nieuwe temperatuur aangehouden zal moeten worden. Vervolgens hoeft de wasmachine taak hier niet meer naar om te kijken. De temperatuur wordt nu volledig door de andere taak gehandhaafd.

Voor de prioritering van de taken is gekeken naar hoe frequent een taak iets moet doen in het systeem. De taken met een clock zijn de Uart, de WashingMachineController, de WebController, de WaterController en de TempController. Aangezien de MotorController geen clock heeft en direct samenwerkt met de WashingMachineController, hebben deze dezelfde prioriteit gekregen, namelijk 0.

De taken zijn hieronder gesorteerd op prioriteit:

- |                            |   |
|----------------------------|---|
| - Uart                     | 0 |
| - WashingMachineController | 0 |
| - MotorController          | 0 |
| - WebController            | 1 |
| - TempController           | 2 |
| - WaterController          | 3 |

De Uart heeft voor zijn clock een tijd van 50 ms. De Uart is daarmee de meest frequente van alle taken en het was dus logisch deze de hoogste prioriteit te geven. De WashingmachineController heeft weliswaar een clock met een tijd van 500 ms, maar is wel de leidende draad van een wasprogramma. Daarom is ervoor gekozen deze (en dus ook de MotorController) dezelfde hoge prioriteit te geven.

Vervolgens was de WebController aan de beurt. Met een clock van 1 seconde is het de minst frequente van alle taken. Echter is ervoor gekozen deze taak toch voorrang te geven



op de WaterController en de TempController, omdat deze de communicatie naar buiten moet verrichten. Het is belangrijk dat deze controller ook echt zo snel mogelijk de beurt krijgt als hij eenmaal de beurt wil.

Vervolgens zijn er nog de taken van de TempController en de WaterController. Deze taken hebben beiden een clock met een tijd van 300 ms. De reden dat toch de TempController een hogere prioriteit heeft gekregen is dat de temperatuur constant kan veranderen als er niks aan wordt gedaan, terwijl (als het goed is) het waterniveau niet zomaar meer verandert als deze eenmaal op pijl is. Daarom leek het handiger de TempController voorrang te geven op het checken van de temperatuur.

## 5.3 Concurrency Model

In het concurrency diagram zijn de verschillende taken binnen het systeem afgebeeld en de manieren waarop ze onderling communiceren weergegeven. De manieren van communicatie tussen de taken die in dit project zijn gebruikt zijn de pool met mutex, de channel, de flag, de clock en de timer.

### 5.3.1 Communicatie middelen

Eerst even een korte beschrijven van de gebruikte communicatie middelen:

Een pool is een manier om data uit te wisselen zonder synchronisatie. Het betreft een enkele waarde, bijvoorbeeld een integer. Het gevaar van een pool is dat de taken die de pool delen tegelijkertijd willen lezen en schrijven in de pool. Om dit te voorkomen kan een mutex op de pool worden gezet. Hiermee kan de gemeenschappelijke toegang tot de pool worden gecoördineerd en conflicten worden voorkomen.

Een channel is een buffer met een vaste grootte waarnaar meerdere taken kunnen schrijven en er slechts één lezer is.

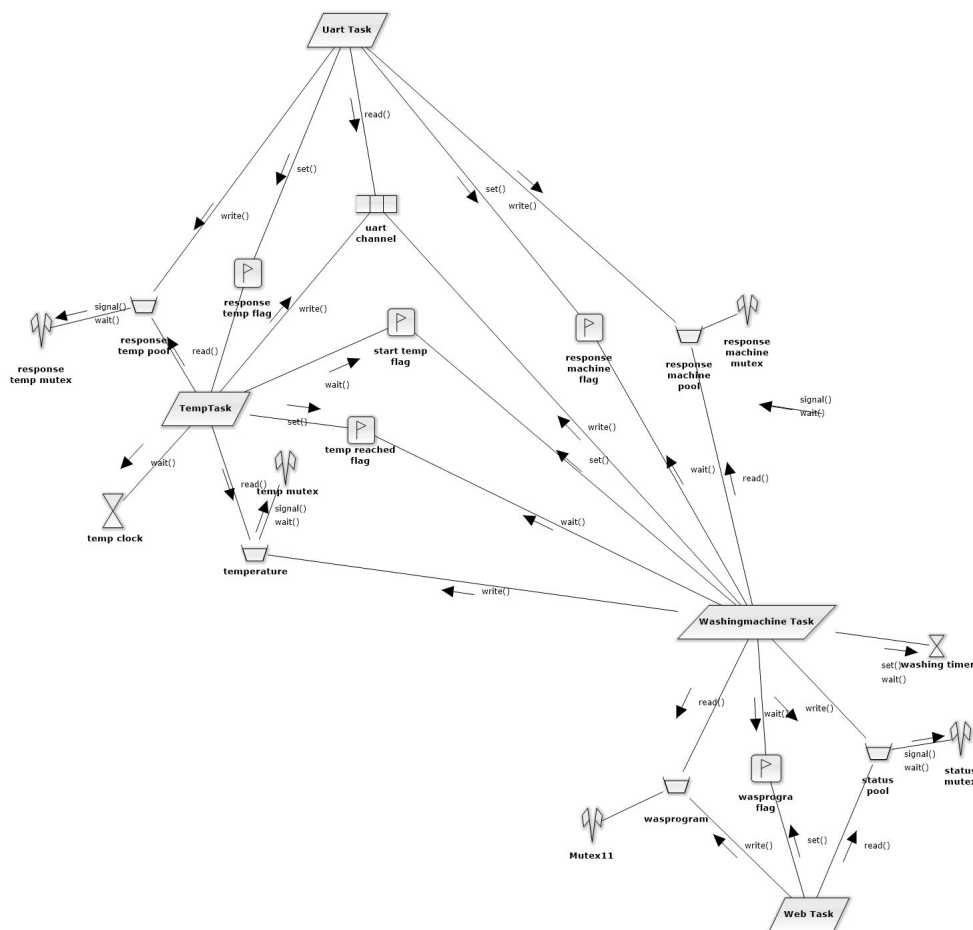
Een flag is als het ware een statusvlag die door een taak kan worden gezet om een andere taak een seintje te geven dat een actie heeft plaatsgevonden. Hierbij vindt verder geen overdracht van data plaats.

Een clock en een timer zijn vormen van tijd-mechanismen. Na enige tijd krijgt de taak die de clock of timer bezit een seintje dat een bepaalde tijd is verstreken. De clock zal telkens na een vaste tijdsduur een seintje geven, terwijl een timer elke keer dat hij is "afgegaan" opnieuw zal moeten worden gezet.

### 5.3.2 Diagram

Het gehele diagram (*bijlagen: VI.I Hele diagram*) ziet er op het eerste gezicht wat rommelig en vol uit. Dit komt omdat alle taken hierin zijn verwerkt.

Aangezien de taken voor de motor, het waterniveau en de temperatuur ongeveer op dezelfde manier communiceren met de Uart, en de WashingMachineController is gekozen ook een versie van het concurrency diagram te maken waarin slechts één van deze drie taken is opgenomen in combinatie met de Uart, de WashingMachineController en de WebController. Aan de hand hiervan (*bijlagen: VI.II Temperatuur deel*) zal de communicatie worden uitgelicht.



De WashingMachineController heeft een eigen taak. Deze taak stuurt naar de Tempcontroller dat hij een bepaalde temperatuur wil hebben door de temperatuur in een pool te zetten die is beveiligd met een mutex. De TempController heeft een clock. Bij het afgaan van deze clock zal de pool worden uitgelezen. De TempController gaat nu regelen dat de temperatuur die in de pool stond wordt bereikt en behouden. Hij werkt daarvoor samen met de tempsensor en de heater. Als de temperatuur die gevraagd was is bereikt zal een flag worden gezet voor de WashingMachineController.

De TempController krijgt van de TempSensor en de Heater een commando dat over de Uart verstuurd moet worden om de wasmachine te laten functioneren. Dit commando wordt

verstuurd via een channel waarin niet alleen de commando's van de TempController komen, maar ook van de andere controllers die communiceren met de Uart.

Als de Uart een commando heeft verwerkt en een antwoord klaar heeft, geeft hij deze direct terug aan de controller waar het commando van afkomstig was door dit in een pool te plaatsen en vervolgens een flag te zetten zodat de controller weet dat het antwoord klaar staat.

De WebController communiceert verder nog met de WashingMachineController. De WebController zal een wasprogramma overdragen aan hem door deze in een pool te plaatsen die is beveiligd met een mutex. Als de WashingController een status terug wil geven, doet hij dit ook door dit in een pool te plaatsen met daarop een mutex.

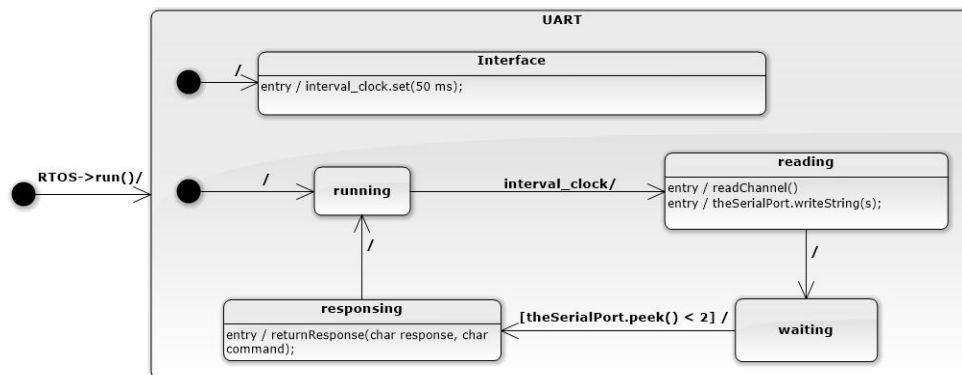
## 5.4 Dynamic Model

In een dynamic model is vastgelegd hoe bepaalde objecten intern functioneren en hoe ze reageren op bepaalde events. Een object kan zich daarbij in een aantal toestanden verkeren en afhankelijk van die toestand zal met een bepaalde actie op een bepaald event worden gereageerd.

Een diagram waarin dit mooi uiteen kan worden gezet is het State Transition Diagram, ook wel STD. Hierin wordt vastgelegd welke events een verandering van toestand veroorzaken en welke acties daarbij uitgevoerd moeten worden.

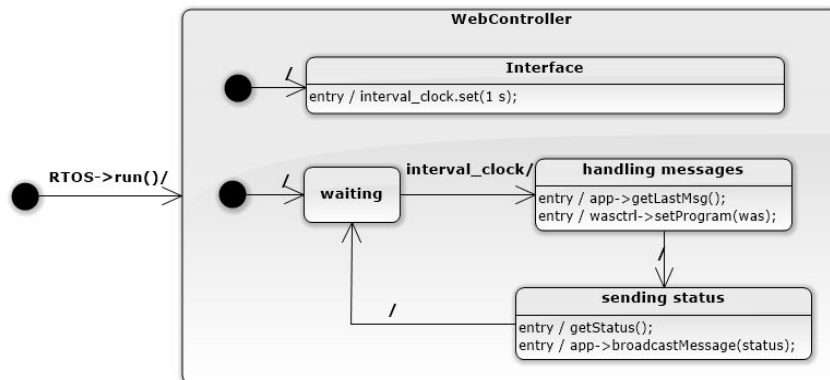
Hieronder zijn voor de taken binnen het systeem verschillende STD's afgebeeld. Er zal per diagram even een korte uitleg volgen.

### 5.4.1 Uart



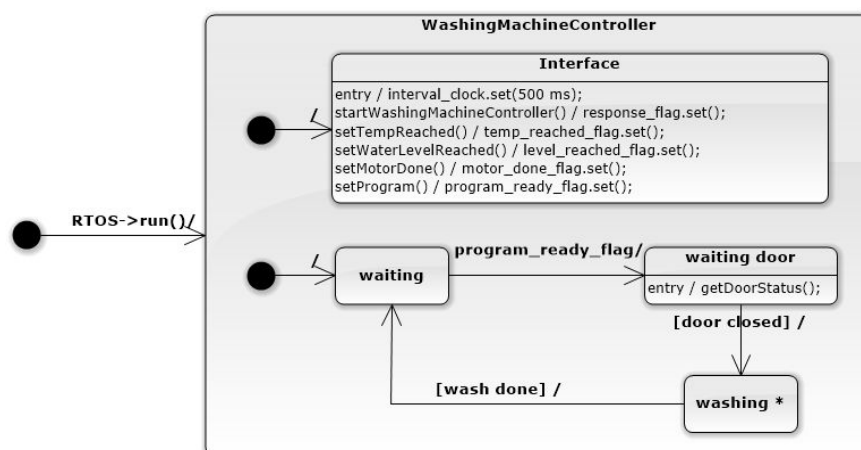
De Uart (*bijlagen: VII.I UART*) kent 4 toestanden waarin hij zich kan verkeren; running, reading, waiting en responding. Zodra de RTOS run wordt aangeroepen zal een interval clock in worden gesteld met een interval van 50 ms. Nu komt de Uart in de toestand running terecht. Hierin is hij actief en wacht op een seintje van de interval clock. Zodra dit seintje binnen is, vindt een verandering van toestand plaats. De Uart is nu in de toestand reading waarin hij een commando uit de channel leest en deze doorstuurt naar over de seriele port. Vervolgens zal hij in de toestand waiting terechtkomen. Hierin blijft hij wachten tot de seriele port een bericht van 2 bytes klaar heeft als antwoord. Nu zal de Uart overgaan naar de toestand responding. In deze toestand zal hij een response bericht sturen naar een andere controller, waarna hij weer terugkomt in de toestand running, om weer te wachten op een signaal van de clock dat hij een nieuw bericht moet verwerken.

## 5.4.2 WebController



De WebController (*bijlagen: VII.II WebController*) kent 3 toestanden waarin hij zich kan verkeren; waiting, handling messages en sending status. Zodra de RTOS run wordt aangeroepen zal een interval clock in worden gesteld met een interval van 1 seconde. Nu komt de WebController in de toestand waiting terecht. Hierin wacht hij op een seintje van de interval clock. Zodra dit seintje binnen is, vindt een verandering van toestand plaats. De WebController is nu in de toestand handling messages waarin hij een wasprogramma opvraagt uit de queue van de webapplicatie en deze vervolgens doorstuurt naar de WashingMachineController. Vervolgens zal hij in de toestand sending status terechtkomen. Hierin zal hij de status van de wasmachine terugsturen naar de webapplicatie. Als dit is gedaan zal worden teruggekeerd in de toestand waiting.

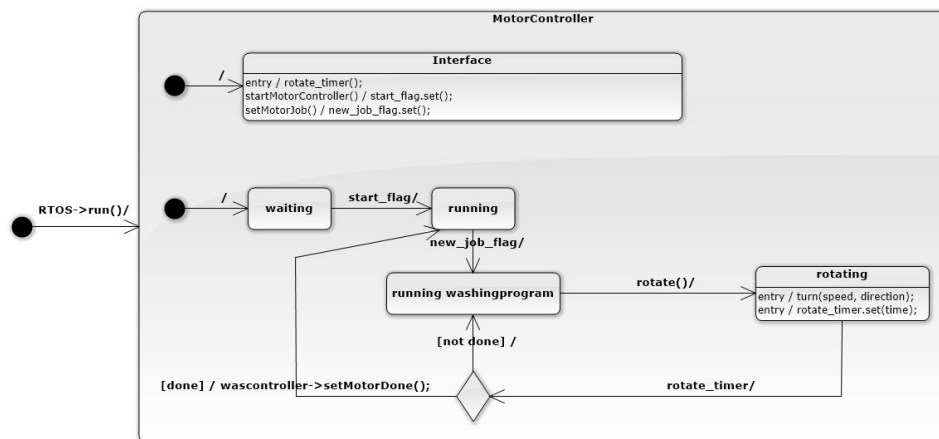
## 5.4.3 WashingMachineController



De WashingMachineController (*bijlagen: VII.III WashingMachineController*) kent 3 toestanden waarin hij zich kan verkeren; waiting, waiting door en washing. Zodra de RTOS run wordt aangeroepen zal een interval clock in worden gesteld met een interval van 500 ms. Nu komt de WebController in de toestand waiting terecht. Hierin wacht hij op het worden geset van de program ready flag. Deze wordt geset door de WebController als het wasprogramma klaarstaat. Zodra dit seintje binnen is, vindt een verandering van toestand plaats. De WashingMachineController is nu in de toestand waiting door, waarin hij wacht tot

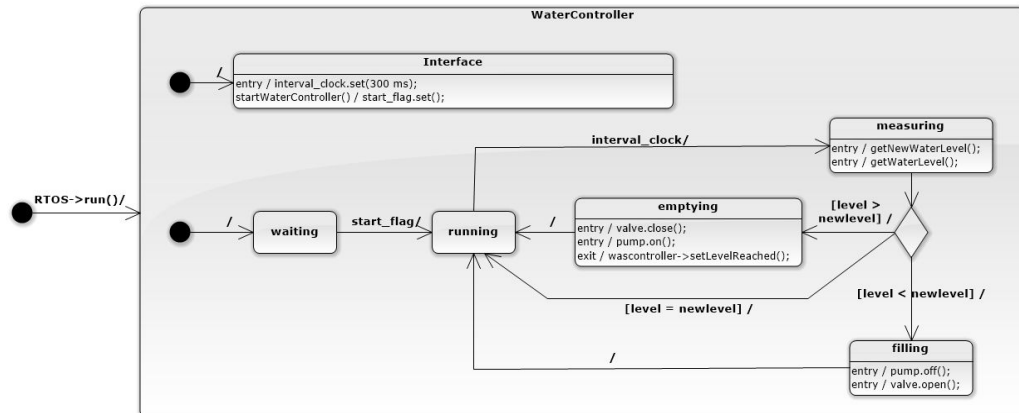
de deur gesloten is. Als de deur is gesloten zal de WashingMachineController in de toestand washing komen. De reden dat hier een sterretje achter staat is dat hierin onderliggend de andere controller aan worden gestuurd voor het regelen van de temperatuur, het waterniveau en het de motor. Als de was als geheel klaar is, keert de WashingMachineController terug in de toestand waiting.

#### 5.4.4 MotorController



De MotorController (*bijlagen: VII.IV MotorController*) kent 4 toestanden; waiting, running, running washingprogram en rotating. Zodra de RTOS run wordt aangeroepen komt de MotorController in de toestand waiting terecht. In waiting wacht de watercontroller op het zetten van de start flag, waarna hij actief mag worden en over zal gaan in de toestand running. In deze toestand zal hij wachten de de flag is gezet die aangeeft dat er een nieuwe job aanwezig is om uit te voeren. Dit programma zal hij gaan draaien in de toestand running washingprogram op de volgende manier. Telkens als de rotate methode aan wordt geroepen zal de MotorController overgaan in de toestand rotating. In deze toestand zal de motor links dan wel rechts gaan draaien en zal een timer worden geset. Bij het afgelopen van deze timer wordt gekeken of de job al klaar is. Is dit het geval dan zal de MotorController terugkeren in de toestand running en wachten op een nieuwe job. Was de job nog niet afgelopen, zal worden teruggekeerd in de toestand running washingprogram waarna nog meer rotaties plaats zullen vinden.

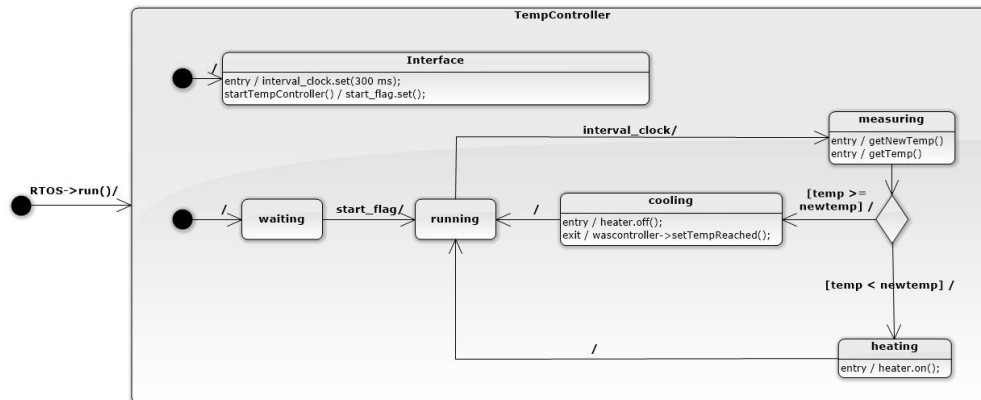
## 5.4.5 WaterController



De WaterController (*bijlagen: VII.V WaterController*) kent 5 toestanden; waiting, running, measuring, filling en emptying. Zodra de RTOS run wordt aangeroepen komt de WaterController in de toestand waiting terecht. Ook is de interval clock op 300 ms ingesteld, hier zal hij in de toestand running naar luisteren. In waiting wacht de watercontroller op het zetten van de start flag, waarna hij actief mag worden en over zal gaan in de toestand running. In deze toestand zal hij wachten op een signaal van de interval clock en nadat dit signaal is gegeven, zal hij overgaan in de toestand measuring. Hierin wordt gekeken wat de nieuwe gewenste temperatuur is en wat de huidige temperatuur is. Deze 2 temperaturen zullen met elkaar worden vergeleken. Vervolgens zal op grond van de uitkomst van deze vergelijken over worden gegaan in de toestand filling, waarin de trommel met water gevuld, of in de toestand emptying, waarin water wordt afgevoerd. Een derde optie is dat het waterlevel precies goed is. In dat geval zal niet eerst de toestand filling of emptying worden aangenomen, maar zal direct worden teruggekeerd in de toestand running.



## 5.4.6 TempController



De TempController (*bijlagen: VII.VI TempController*) kent 5 toestanden; waiting, running, measuring, heating en cooling. Zodra de RTOS run wordt aangeroepen zal de interval clock worden ingesteld op 300 ms, ook komt de tempcontroller in de toestand waiting terecht, waarin hij wacht op de start flag. Zodra de start flag is geset, komt de TempController in de toestand running terecht. Als vervolgens de interval clock een signaal geeft, zal worden overgegaan naar de toestand measuring. In deze toestand zal de nieuwe gewenste temperatuur worden gelezen en worden gekeken wat de huidige temperatuur is. Deze zullen met elkaar worden vergeleken. Nu vindt een splitsing plaats. Mits de gewenste temperatuur hoger is dan de huidige temperatuur, zal worden overgegaan naar de toestand heating, waarin de verwarming aan wordt gezet. Hierna komt de TempController weer in de toestand running terecht. Als de gewenste temperatuur lager is dan de huidige temperatuur zal in plaats van de toestand heating, de toestand cooling worden aangenomen. Hierin wordt de verwarming uitgezet en krijgt de WashingMachineController een signaal dat de temperatuur warm genoeg is. Hierna zal ook de toestand running weer worden aangenomen.

## 6 Webapplicatie

Voor dit project was het de opdracht om de wasmachine te kunnen aansturen via een webapplicatie, die de was kan instellen en dan kan starten of zelfs stoppen als het nodig mocht zijn.

### 6.1 Eisen

Er zaten natuurlijk ook een aantal eisen aan de webapplicatie gebonden, die door de klant waren aangegeven. Vooral qua vormgeving. Het was de bedoeling dat de gebruiker geen moeite hoeft doen om te snappen hoe de webapplicatie werkt. De webapplicatie moest dus heel overzichtelijk zijn.

### 6.2 Keuzes

Er is gekozen om de webapplicatie voor de webbrowser op je PC te maken. Er is hiervoor gekozen omdat in de groep weinig ervaring was met het maken van een website. Dit moest dus geleerd worden. Voor de PC een webapplicatie maken leek het makkelijkste en daarnaast het meest bereikbare voor de gebruiker, Want iedereen heeft wel een PC thuis staan.

Het doel van de webapplicatie is dat je vanaf de webbrowser je wasmachine kan aansturen. dus wij hebben ervoor gekozen dat je gelijk op de pagina komt met de opties om de was in te stellen.

Maar voordat je daar komt moet je wel eerst inloggen. Er is daarvoor gekozen omdat het niet fijn zou zijn als iedereen zomaar jou wasmachine zou kunnen aan of uitzetten. Vandaar een klein inlogscherms waar je je gebruikersnaam een wachtwoord in moet voeren.

Bij het instellen van de was is de keuze gemaakt dat de gebruiker kan kiezen uit een aantal opties: Hoeveel water de wasmachine gebruikt, Welke type was de gebruiker wil draaien, op welke temperatuur de was wordt gedraaid en hoelang de was wordt gedraaid. Wanneer de "Was type" wordt geselecteerd dan kiest de applicatie zelf welke instellingen het beste zijn voor dat type was.

Er is daarnaast ook nog gekozen dat de was niet gestart kan worden wanneer niet alle vakjes ingevuld zijn. Wanneer toch op de startknop wordt gedrukt worden de vakken die nog niet waren ingevuld rood aangegeven.

Er is ook gekozen om de heater met daarop de tijd, status, stopknop en resterende tijd altijd weer te geven. het maakt niet uit op welke tabblad de gebruiker zich bevindt. Het leek verstandig om altijd te zien wanneer de was klaar zou zijn en wanneer er een noodstop gedaan moet worden zou de gebruiker niet eerst van tabblad willen veranderen.

Bij de navigatie van de webapplicatie is gebruik gemaakt van JQuery. Hierdoor kon gelijk de hele pagina geladen worden. Waardoor niet meer geladen hoeft te worden tussen de tabs door.

Er is ook nog gekozen om een uitlogknop te maken. wanneer hier op wordt gedrukt is er gekozen om een pop-up te laten verschijnen waarop wordt gevraagd of je zeker weten wil uitloggen.

Hierbij zijn de keuzes van de webapplicatie wel duidelijk gemaakt.

# 7 Realisatie

## 7.1 Werkomgeving

Om het systeem te ontwikkelen zijn verschillende tools en applicaties gebruikt: Software Ideas Modeler (SIM), Visual Studios 2015 (VS15), Notepad++, Microsoft Office, Google Docs (GDoc), makefiles, C++, Putty, Filezilla, HTML5, Doxygen, JavaScript en CSS. Per onderdeel van het eindproduct, zal hieronder een korte omschrijving volgen van de gebruikte tools en werkomgevingen.

### 7.1.1 De website

Voor het maken van de website is de browser Google Chrome gebruikt en is geschreven in de taal HTML5 met JavaScript en CSS ondersteuning. HTML5 is de opkomende nieuwe standard voor webbrowser taal voor de layout van een pagina. CSS is een ondersteunende configuration taal om bij het maken van de layout te helpen, door bepaalde layout stukken af te scheiden van de html5 code. Hiermee wordt de content en de format beter gescheiden. Vergelijkbaar zoals bij Tex - maar dan interpreted based inplaats van compile based. JavaScript is een interpreted taal die als ondersteuning bied voor HTML5 door functies (zoals in C++ en Java) door de browser te laten uitvoeren i.p.v door de server waar de pagina vandaan komt. Als je op een knop drukt dan hoeft er niet tegen de server verteld te worden dat dit gebeurt is. Dit bespaard tijd en performance op de server.

### 7.1.2 De modellen

Voor het maken van de modellen is gebruik gemaakt van SIM. met SIM kunnen een grote hoeveelheid verschillende diagrammen gemaakt worden zoals de belandrijke Use Case, Activity, Constraints, Concurrency, Class en State Transition Diagrams. Maar ook flowcharts en sequence diagrams.

### 7.1.3 De code ontwikkelomgeving

De code is ontwikkeld in VS15 en Notepad++ in de taal C++. VS15 is een volledige Integrated Development Enviroment (IDE). Dat wil zeggen; Alles wat je nodig hebt om te programmeren is in één programma samengevoegt. Compileren en Linken gebeurt met één druk op de knop, functies worden automatisch samengevoegt en compiler errors worden aangegeven vóór het compileren en verschillende geadvanceerde functies zoals een virtueel filesysteem of het maken van een klassendiagram van de code. VS15 is echter erg zwaar, dus daarom wordt soms enkel Notepad++ gebruikt. Een geadvanceerde textverwerker die uiteindelijk alleen maar syntax hilightning heeft.

VS15 is echter een windows-only IDE en heeft compatibiliteits problemen met de Unix OSen zoals Raspbian of Dabian. Dit komt omdat de system calls in Unix vrij zijn aan te roepen (zonder includes) en bij windows moet er eerst een include worden gebruikt. Includes voor UNIX systemcalls op windows zijn echter niet beschikbaar. Daarom wordt de code alleen

geschreven in VS15 maar wordt het programma gebuild (gecompileerd en gelinkt) door middel van een handgemaakte makefile (VS15 doet dit normaal automatisch) die op de raspberry wordt gebruikt..

C++ is de Object-Oriented (OO) variant van C. Hoewel C++ een grotere footprint heeft (een c++ programma is groter op het RAM dan een C programma), maakt het een stuk makkelijker om grote programmas te schrijven met de relatieve snelheid van C, maar met het schaalvoordelen en organisatie gemak van bijv. Java.

Voor de documentatie is gebruikt van DoxyGen. DoxyGen is een inline code die de C++ code leest en daar een webpagina in HTML maakt of code voor een (La)Tex bestand. Dit is een geautomatiseerde manier om geordende documentatie te maken zoals op websites zoals Cplusplus.com te zien is (HTML) of als een mooi PDF (Latex)..

### 7.1.3 Raspberry PI

De raspberry heeft een sd kaart waar het OS op komt te staan. De gebruikte linux distributie is debian jessy, dit werd geïnstalleerd op het sd kaartje. Vervolgens konden we de sd kaart in de raspberry pi steken en opstarten.

Om te kunnen verbinden met de raspberry pi hebben we gebruik gemaakt van het programma putty. De eerste keer dat de raspberry pi werd opgestart hebben we gebruik gemaakt van de COM poort om te verbinden en in te loggen op de raspberry pi. Via deze verbinding is de ethernet verbinding opgesteld zodat er nu ook verbonden kon worden via de ethernet kabel. Als de raspberry werd verbonden met een laptop kon het internet gedeeld worden zodat de raspberry pi ook internet had om programma's te kunnen installeren.

Later in het project waren er ook momenten dat meerdere mensen op de raspberry pi moesten, omdat hiervoor gebruik werd gemaakt van de directe ethernet verbinding kon maar een persoon verbinden, dat was niet altijd even handig omdat er constant gewisseld moest worden. Om dit op te lossen is er een router aangesloten met wifi. Hierdoor kon iedereen met de raspberry pi verbinden via het wifi netwerk van die router. Zo kon iedereen verbinden via ssh om zijn deel op de raspberry pi te doen.

Om de code en website files op de raspberry pi te krijgen is het programma filezilla gebruikt. Dit programma kan via het protocol sftp verbinden met de raspberry pi. Dit geeft ons toegang tot alle files. Echter heeft de verbinding via sftp niet alle rechten om in alle files te schrijven. Dit kan aan via het linux commando chown. Hierna konden we alle files direct op de raspberry pi zetten.

Voor het laden van de website hebben we op de pi een webserver draaien. De webserver die gebruikt is nginx. Voor deze webserver in combinatie met de raspberry pi was op het internet al wat documentatie beschikbaar waardoor het installeren van de server snel en eenvoudig te doen was. Het vervolgens werkend krijgen van de websites was nog wel wat lastig omdat hij niet gelijk de bestanden voor de website wou laden. Door wat aanpassingen te maken in wat instellingen hebben uiteindelijk alle benodigde files in kunnen laden.

## 7.2 Uitzonderlijke code

Een ding die voor problemen zorgde was de cyclische depandency die in het ontwerp te vinden was. Hierbij ging het om de taken die elkaar moeten kennen om te kunnen communiceren met elkaar. De WashingMachineController moet de WaterController, de TempController en de MotorController kennen om te kunnen communiceren, maar de communicatie moet in beide richtingen verlopen, dus deze controllers moeten ook de WashingMachineController kennen.

Hierdoor ontstond een probleem want de ene klas include ook de andere klas en beide klassen worden in beide gebruikt. Dus dan kent de compiler de klas niet omdat hij hem aan het compilen is, om dit op te lossen kan een klas voorwaarts gedeclareerd worden. Dit houdt in dat bovenaan in een klasse een lege class wordt neergezet met de juiste naam. Dit laat aan de compiler weten dat die class echt bestaat en gewoon kan gebruiken. Als nu echter de include wordt weggelaten kunnen de functies niet gebruikt worden omdat hij dan alleen die lege voorwaarts gedeclareerde klasse kent. Het is dus noodzakelijk om zowel de voorwaartse declaratie te doen als de include.

## 8 Evaluatie

In dit hoofdstuk wordt besproken wat er tijdens dit project niet zo goed ging als er op het begin gehoopt was. En hoe we dit hebben opgelost om het toch tot een succesvol project te leiden. En hoe dit in een volgend project beter zou kunnen.

### 8.1 Problemen & Oplossingen

Ten eerst was er veel te veel op het bordje genomen. Van een aantal taken was er verwacht dat het wel zo gedaan had kunnen worden. Hierdoor is het product niet zo vlekkeloos geworden als op het begin gehoopt was. Dit is zo goed mogelijk opgelost door sommige taken minder prioriteit te geven, zodat er wel een werkend product wordt opgeleverd maar deze niet heel veilig of mooi is. De volgende keer als er een project wordt gestart is het dan handig om minder taken in de Must van de MoSCoW te zetten en ook beter de MoSCoW te volgen.

Ten tweede was er op het begin een beetje last van weinig communicatie in het groepje. Hierdoor waren niet alle taken die ingeleverd moesten worden op orde. Dit was eigenlijk al heel snel verholpen toen we erachter kwamen. Hierna hadden we alles op tijd ingeleverd en was er veel meer communicatie.

Nog een probleem waar tegenaan is gelopen tijdens het opzetten van de wasmachine, was een probleem met de raspberry pi. Hierop internet toegang verkrijgen zonder de pi direct op het internet aan te sluiten bleek niet zo makkelijk als gehoopt. Uiteindelijk is dit nog wel gelukt, maar het heeft een aardige tijd in beslag genomen.

## 9 Conclusies en aanbevelingen

Terugkijkend op het project vonden we het een leuke uitdagende opdracht. Een punt voor de volgende keer is wel dat er iets meer tijd beschikbaar komt om het project goed af te ronden. Nu hadden we het idee dat (mede door wat tegenslagen binnen het team) op het einde met veel haast gewerkt moest worden om toch nog alles zo goed mogelijk af te krijgen. Dit is jammer, want met een weekje extra tijd had het eindresultaat er een stuk beter uitgezien.

Voor het maken van het systeem is duidelijk geworden hoe erg het maken van de verschillende modellen van belang zijn voor het schrijven van de code. Als deze redelijk op orde zijn, is het maken van de code een stuk beter te behappen. Wel was dit voor ons ook een leerproces en bleek uiteindelijk dat bepaalde dingen in ons klassendiagram bijvoorbeeld toch niet makkelijk te verwezenlijken bleken. Dit hebben we dan ook aan moeten passen.

Al met al was het een uitdagend en leerzaam project.



# 10 Bronvermeldingen

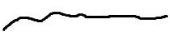












- [1] Van Ooijen, W., Wensink, M. (2013-02-05). Realtime System Programming.
- [2] Van Ooijen, W. slides C++ programming & SE2.
- [3] Wensink, M. (2015, 5 november). Beschrijving wasmachine-emulator.
- [4] Van Dijk, S. (2015, 26 augustus). Handig! Alle wasprogramma's voor een wasmachine bij elkaar. Retrieved from <http://www.greenem.nl/wasprogrammas-voor-een-wasmachine/#1440592086498-789a26f6-22e2>
- [5] Organisatie Siemens. Retrieved from [http://www.wehkamp.nl/privatedata/pdf/403347\\_PDF.pdf](http://www.wehkamp.nl/privatedata/pdf/403347_PDF.pdf)
- [6] Organisatie Exquisit. (2012, november). Wasmachine WA5514 A+ WA6514 A+ WA7514 A++. Retrieved from [http://www.domest.nl/uploads/catalogerfiles/wa7514/Handleiding\\_Wasmachine\\_WA\\_7514\\_A\\_Nederlands\\_20121106.pdf](http://www.domest.nl/uploads/catalogerfiles/wa7514/Handleiding_Wasmachine_WA_7514_A_Nederlands_20121106.pdf)
- [7] organisatie w3schools. (1998 - heden). retrieved from <http://www.w3schools.com/>
- [8] Setting up an NGINX web server on a Raspberry Pi. (n.d.). Geraadpleegd op Januari 25, 2016, from <https://www.raspberrypi.org/documentation/remote-access/web-server/nginx.md>
- [9] Cplusplus.com - The C Resources Network. (2000- heden). Geraadpleegd op Januari 25, 2016, from <http://www.cplusplus.com/>

# Bijlagen

## I MosCow

### must:

- Een webapplicatie moet de wasmachine via de "raspberry pi" aansturen. Dit houdt bijvoorbeeld het starten van een wasprogramma, de was logs opvragen en eventueel het stoppen van het wasprogramma in.
- Webapplicatie gegarandeerd compatibel met een desktop/laptop dat een webbrowser ondersteunt.
- Bij hardware fouten moet een wasprogramma meteen worden afgebroken/ gestopt en überhaupt niet mag starten. Een hardware fout is een kapot onderdeel.
- Bij een fout de gebruiker grafisch informeren via de webapplicatie.
- Een wasprogramma met een pop-up notificatie.
- Wanneer noodstop-knop wordt ingedrukt, wordt het wasprogramma afgebroken en het water stroomt zo snel mogelijk weg. Daarna kan de deur worden geopend.
- De wasmachine moet internetonafhankelijk het wasprogramma kunnen afmaken.
- Simplistisch Grafical User Interface (GUI) - alleen een hoofdpagina met enkele knoppen om te starten en duidelijke plaatjes en teksten.

|  |   |
|--|---|
| <b>Wasmachine</b>  | <b>13:00</b> <b>status: spoelen</b> <b>S</b> <b>resterend: 01:34</b>  |
| <br><br><br> | <b>favorieten:</b>     |
|  |     |
|  |     |
|  | <b>START</b>  |

- Inloggen met een wachtwoord om op de webapplicatie toegang te krijgen.
- De status waarin de wasmachine verkeerd. dus de resterende tijd en de status waarin de wasmachine zich verkeerd moet altijd up-to-date zijn in de webapplicatie, ook na het wegvallen van de internetverbinding. Zodra de verbinding is hersteld moet zo snel mogelijk de meest recente status weer op worden gevraagd.

should:

- Logs bijhouden en kunnen weergeven van energie- en waterverbruik, meest gebruikte wasprogramma.
- Vier (of meer) wasprogramma's.
- Status van het wasprogramma is altijd zichtbaar op de webpagina. dit zijn de punten die worden weergegeven: De resterende tijd, wat de wasmachine op het huidige moment aan het doen is en de huidige temperatuur van het water in de wasmachine.
- Software van de wasmachine moet updatebaar zijn door middel van het internet. De gebruiker wordt op de webapplicatie gevraagd of hij/zij een update wil uitvoeren.
- Uitgebreid GUI - meerdere tabs voor een mode voor alleen de monteur of waar je al het verbruik van de wasmachine goed in kunt terug vinden. waarin we geen scroll-schermen in gaan gebruiken.

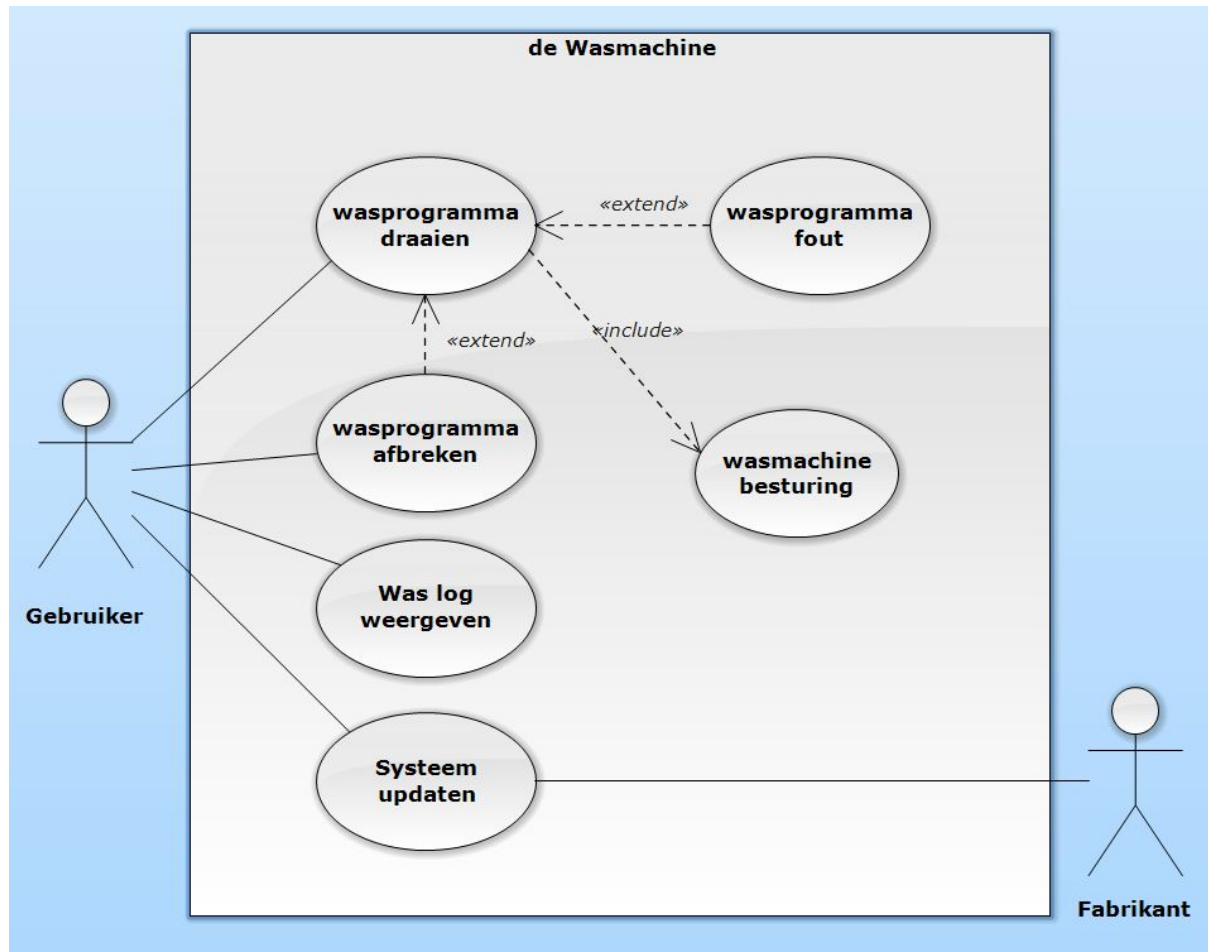
could:

- Bij slijtage/schade - op toestemming van de klant - bericht naar de fabrikant sturen dat het onderdeel niet meer goed is.
- De webapplicatie is gegarandeerd compatible met webbrowsers Chrome, Safari en Firefox.
- De webapplicatie is gegarandeerd compatibel met desktop, tablet, smartphone.
- Van te voren in kunnen plannen wanneer het wasprogramma start.
- Luxe uitvoering van de GUI: visual effects (animaties) toevoegen aan de webpagina.

would like but won't get:

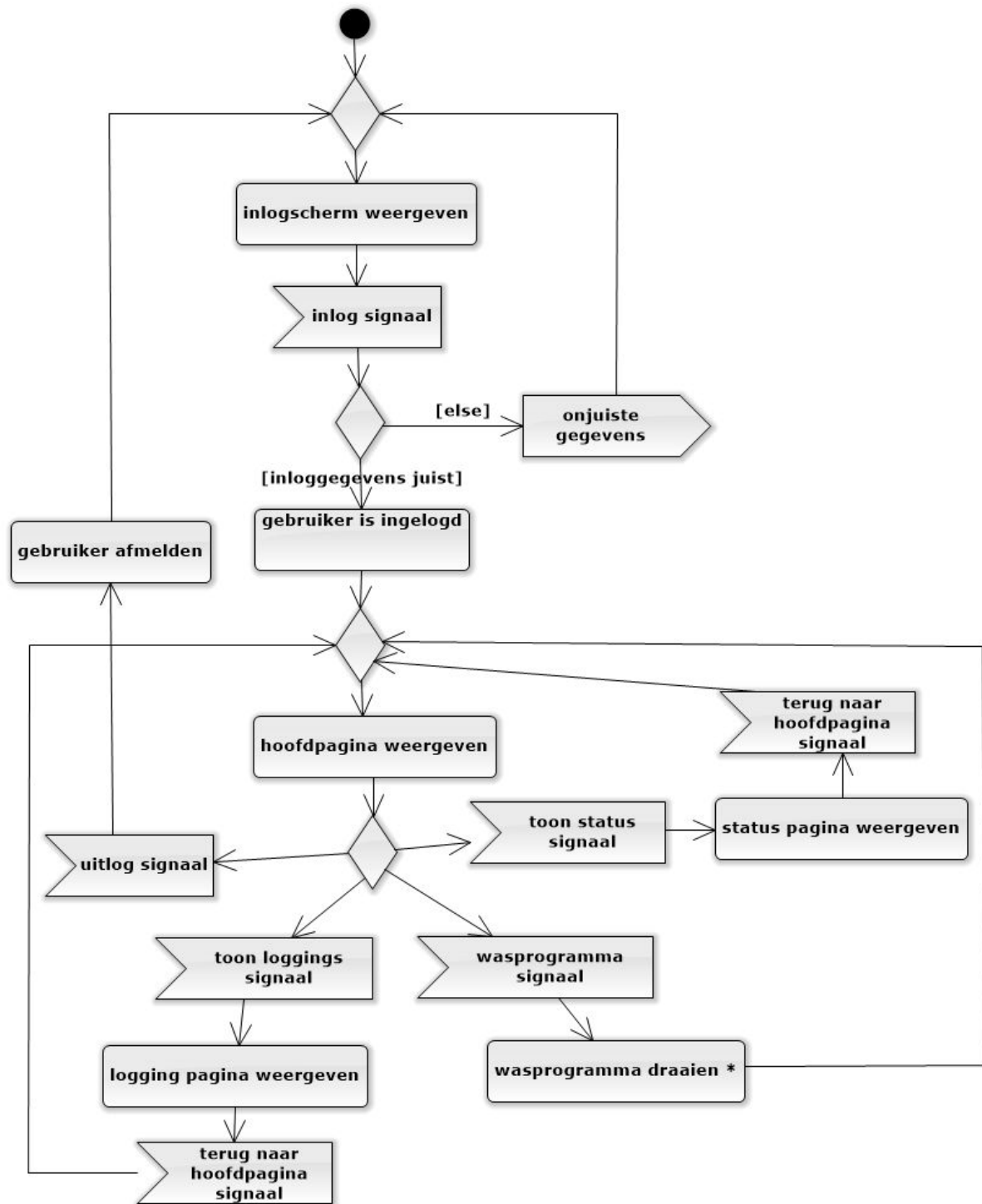
- De webapplicatie laten werken met meerdere wasmachine's (wasserette).
- ook andere programmeurs een wasmachineaanstuurprogramma kunnen maken op basis van het onze.

## II Use Case Diagram

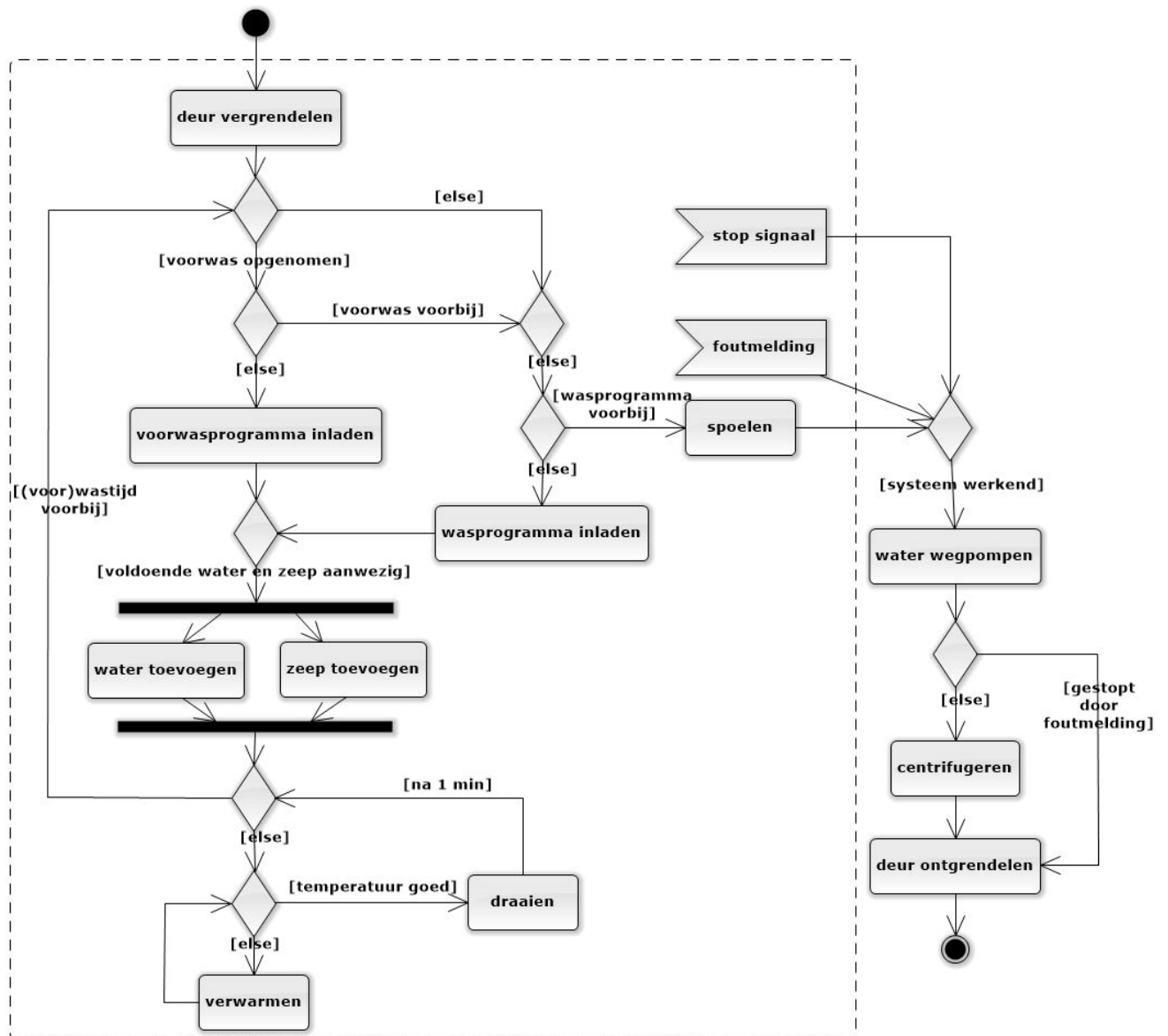


## III Activity Diagrams

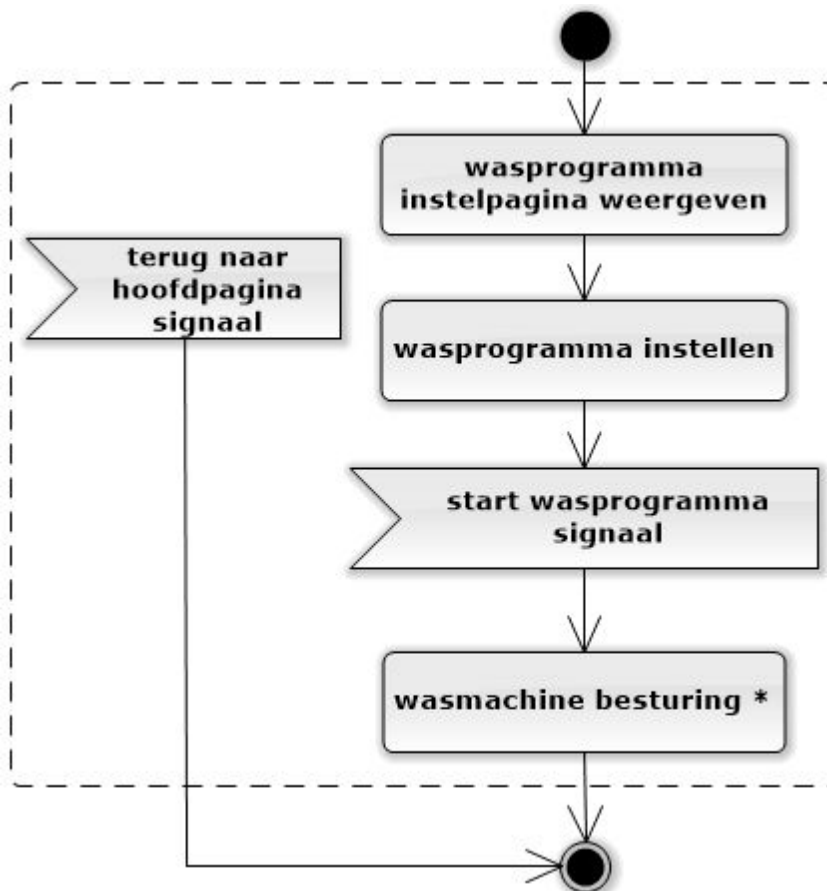
### III.I User Interface



### III.II Wasmachine Besturing



### III.III Wasprogramma Draaien



## IV Constraints Model

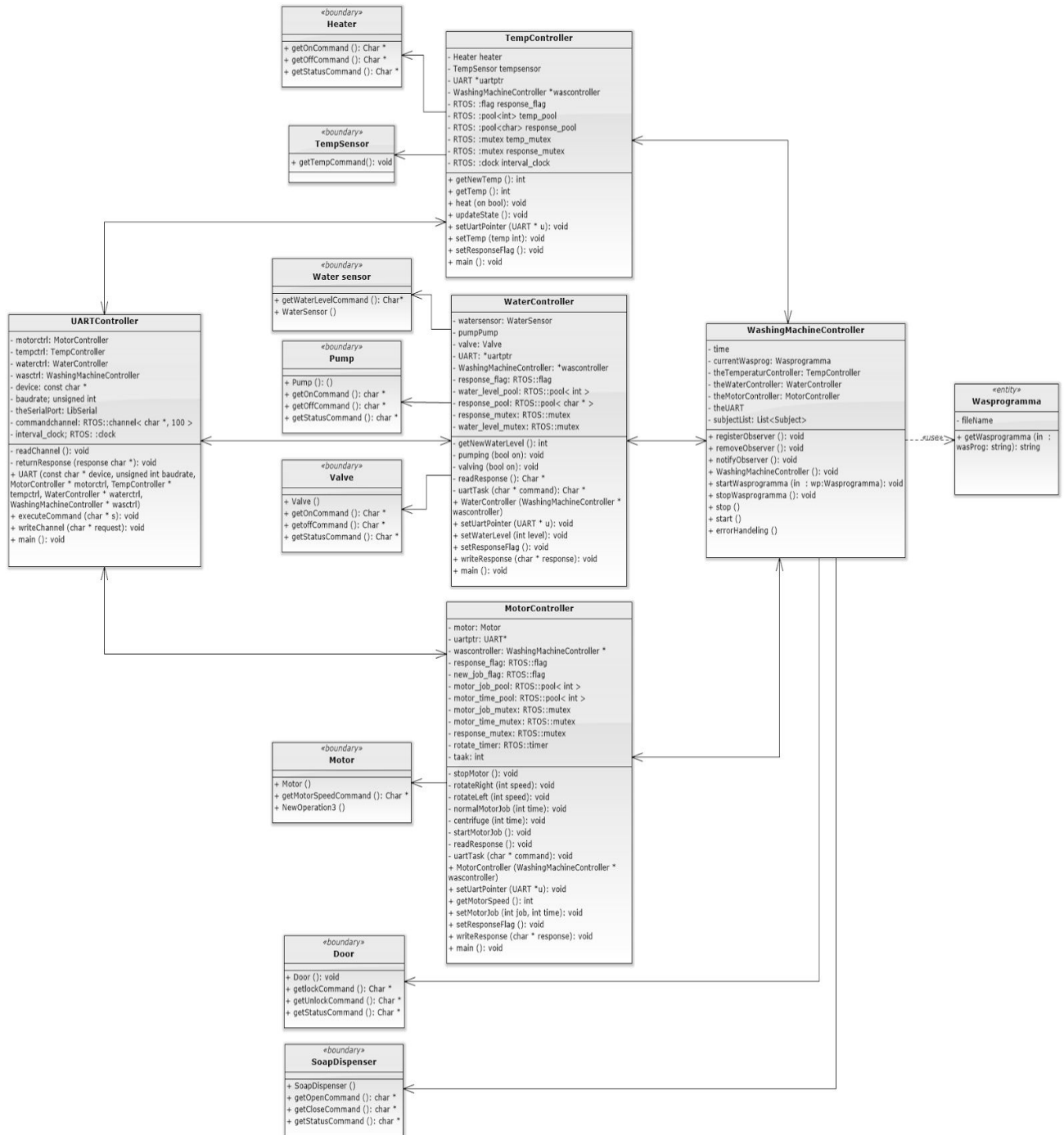
| Constraint type | Identificatie                            | Beschrijving  | Criterium   | Verificatie methode   |
|-----------------|--|---|---|---|
| Performance     | Reactietijd server                       | De tijd waarin de server moet reageren op requests van de webapplicatie.                                  | Als er een knop op de webpagina wordt geklikt, mag de reactie van de server 60 seconden later zijn.         | Als er op de knop gedrukt is, meten hoelang het duurt voordat de server reageert. Zowel bij een bekabelde verbinding als een wifi verbinding. En kijken wanneer de wasmachine gaat draaien. |
| Performance     | Reactietijd webapplicatie                | De tijd waarin de webapplicatie reactie geeft bij het uitvoeren van iets, zoals het drukken van een knop. | Als er een knop op de webapplicatie wordt geklikt, mag de reactie 0,5 seconden later zijn.                  | Als er op de knop gedrukt is, meten hoelang het duurt voordat de webapplicatie reageert. Zowel bij een bekabelde verbinding als een wifi verbinding.  |
| Performance     | Button uitlezen                          | Tijd waarbinnen een button event moet worden afgehandeld  | Binnen 20 ms  | Door middel van debug-code die aangeeft hoeveel tijd er zit tussen het event en het hebben afgehandeld van dat event.   |
| Performance     | Temperatuur uitlezen en heater aansturen | Tijd waarbinnen de temperatuursensor uitgelezen moet worden + de heater aangestuurd moet worden           | Binnen 50 ms  | Door middel van debug-code die aangeeft hoeveel tijd er zit tussen het uitlezen van sensor en het aan/uitzetten van het verwarmingselement.   |
| Accuracy        | Temperatuur afwijking                    | De afwijking in temperatuur   | De temperatuur mag niet meer dan 3°C afwijken.  | De temperatuur meten terwijl er verschillende wasprogramma's draaien.   |
| Accuracy        | Tijd afwijking webapplicatie             | De hoeveel tijd die de klokken op de webapplicatie mag afwijken   | De tijd mag uiterlijk niet meer dan 1 minuut afwijken en liever niet meer dan 30s.                          | De timer een erg lange tijd laten lopen en kijken hoeveel de tijd afwijkt.  |
| Resource Use    | Energie gebruik                          | Hoeveelheid Energiegebruik  | Het systeem mag niet meer dan 1 kWh gebruiken   | Meten met een multimeter tijdens verschillende stadia van verschillende wasprogramma's  |
| Resource Use    | Water gebruik                            | Hoeveelheid watergebruik  | Het systeem mag niet meer dan 40 liter water gebruiken per wasbeurt.  | De output aansluiten op een maatemmer en meten hoeveel maatemmers er nodig zijn.  |
| Resource Use    | RAM gebruik                              | Hoeveel RAM de applicatie nodig heeft op de Raspberry Pi  | Het programma mag niet meer 256MB RAM gebruiken at-runtime  | Over een bepaalde tijd het RAM gebruik meten van de applicatie at-runtime.  |
| Availability    | Beschikbaarheid van de webapplicatie     | Wanneer is de webapplicatie beschikbaar?  | De webapplicatie moet beschikbaar zijn als er internetverbinding bestaat tussen de webserver en de browser. | Fysiek internetkabel tijdelijk los maken en daarna vastmaken en daarbij de browser verversen..  |
| Reliability     | Betrouwbaarheid van de webapplicatie     | Beschikbaarheid van het systeem.  | Mits er internet connectie is en er stroom is, moet het systeem 99% van de tijd werken.                     | De wasmachine zoveel mogelijk uren laten maken en het uitvallen meten.  |



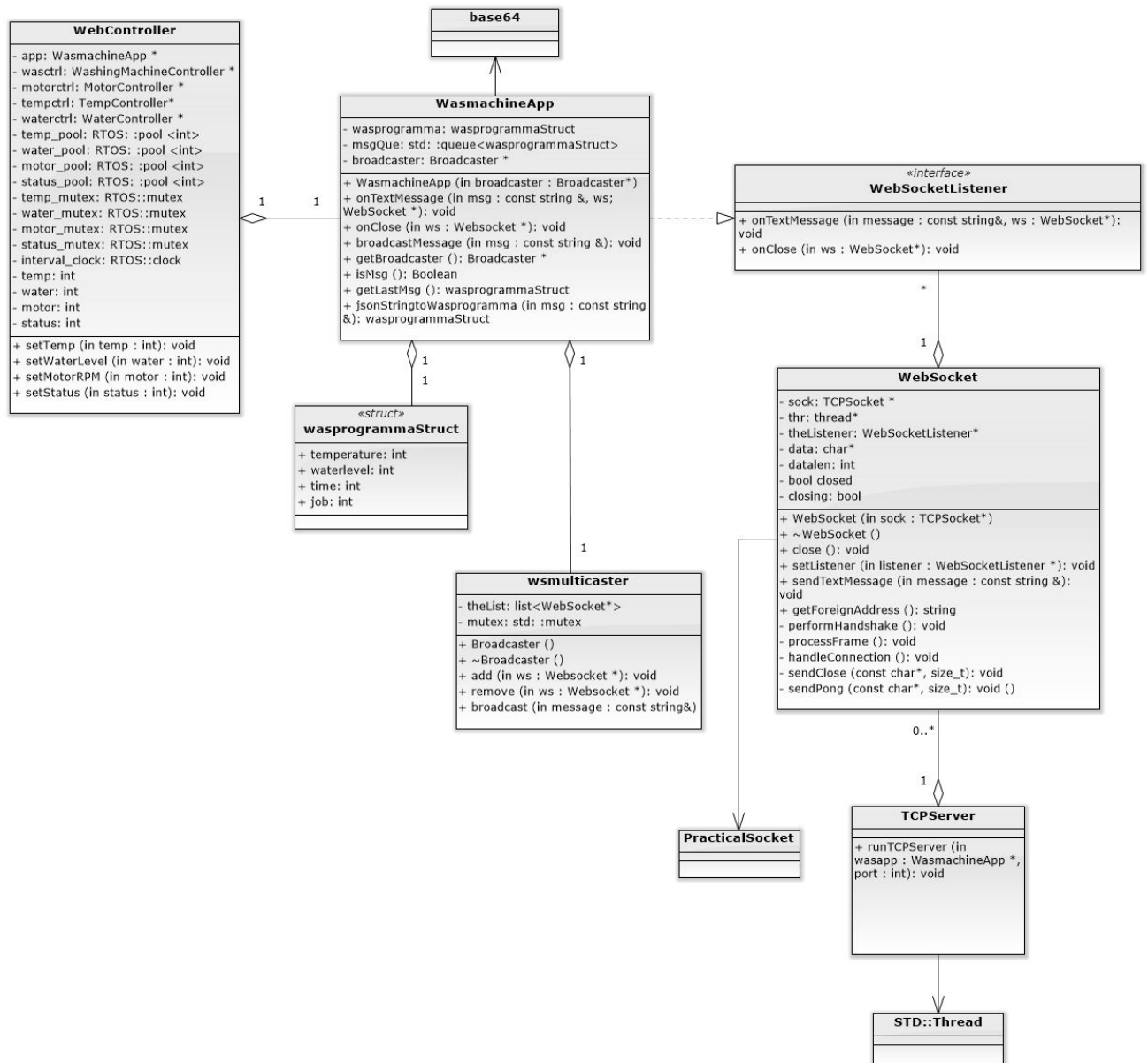
|              |                    |   |   |   |
|--------------|--------------------|---|---|---|
| Robustness   | Falen van hardware | Wat te doen als een onderdeel faalt   | Als een onderdeel faalt dan moet het wasprogramma zodanig worden gestopt dat er geen extra schade ontstaat. | Onderdelen verwijderen/loskoppelen om 'kapot gaan' te simuleren en kijken hoe het systeem er op reageert. |
| Robustness   | Internet ontbreken | De essentie van toegang tot het internet voor de wasmachine                         | De wasmachine moet het wasprogramma kunnen afmaken, ookal valt het internet uit.                            | Fysiek de internetkabel los maken van de wasmachine en kijken of het wasprogramma goed doorgaat..         |
| Learnability | Gebruiksgemak      | Kan de gebruiker de webapplicatie gebruiken zonder de handleiding te hebben gelezen | Nee, de gebruiker hoeft de handleiding niet gelezen te hebben om de webapplicatie te kunnen gebruiken.      | Iemand die niet de handleiding heeft gelezen de was laten doen.   |

# V Klassendiagram

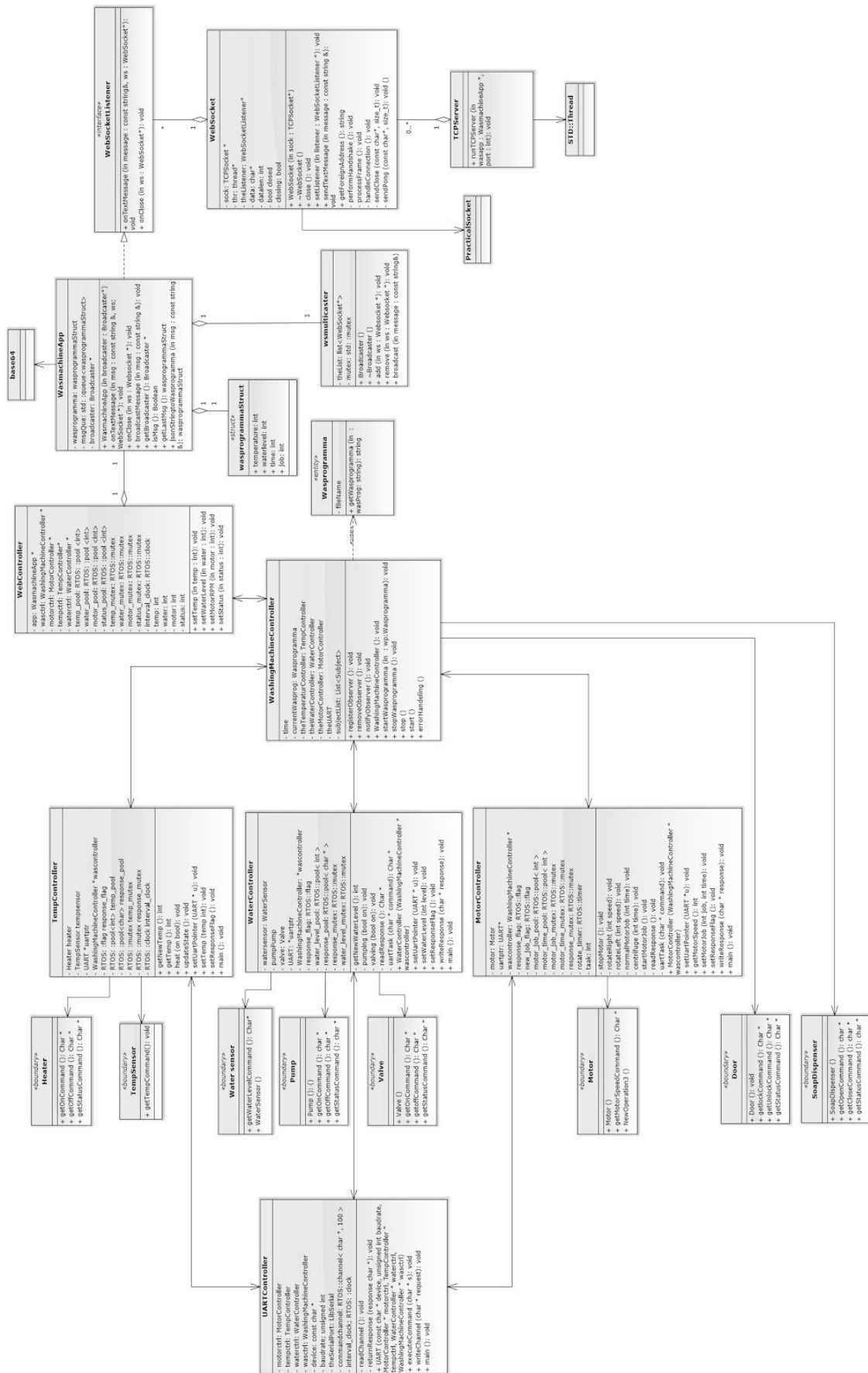
## V.I Wasmachine deel



## V.II Webapplicatie deel

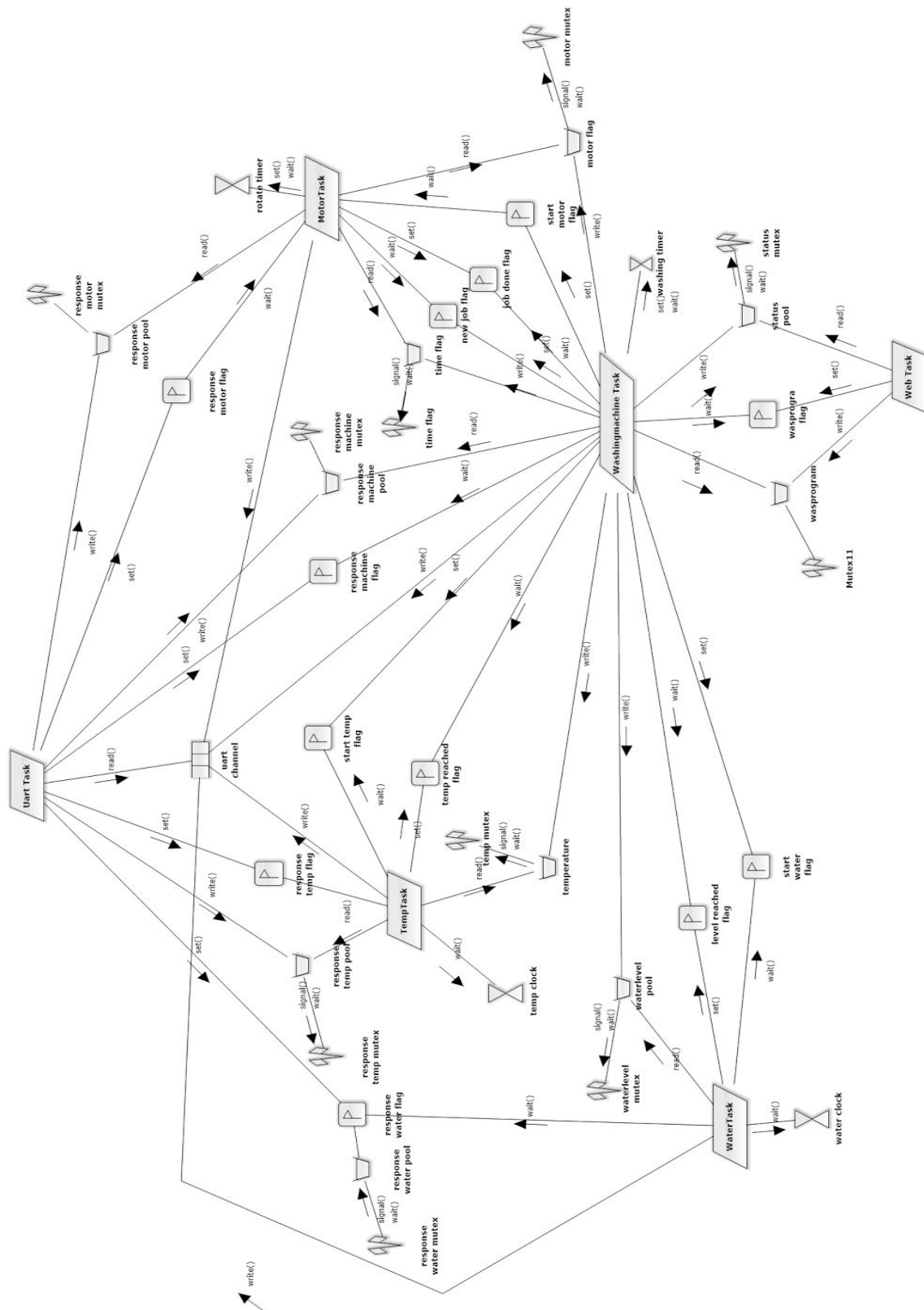


### V.III Hele diagram

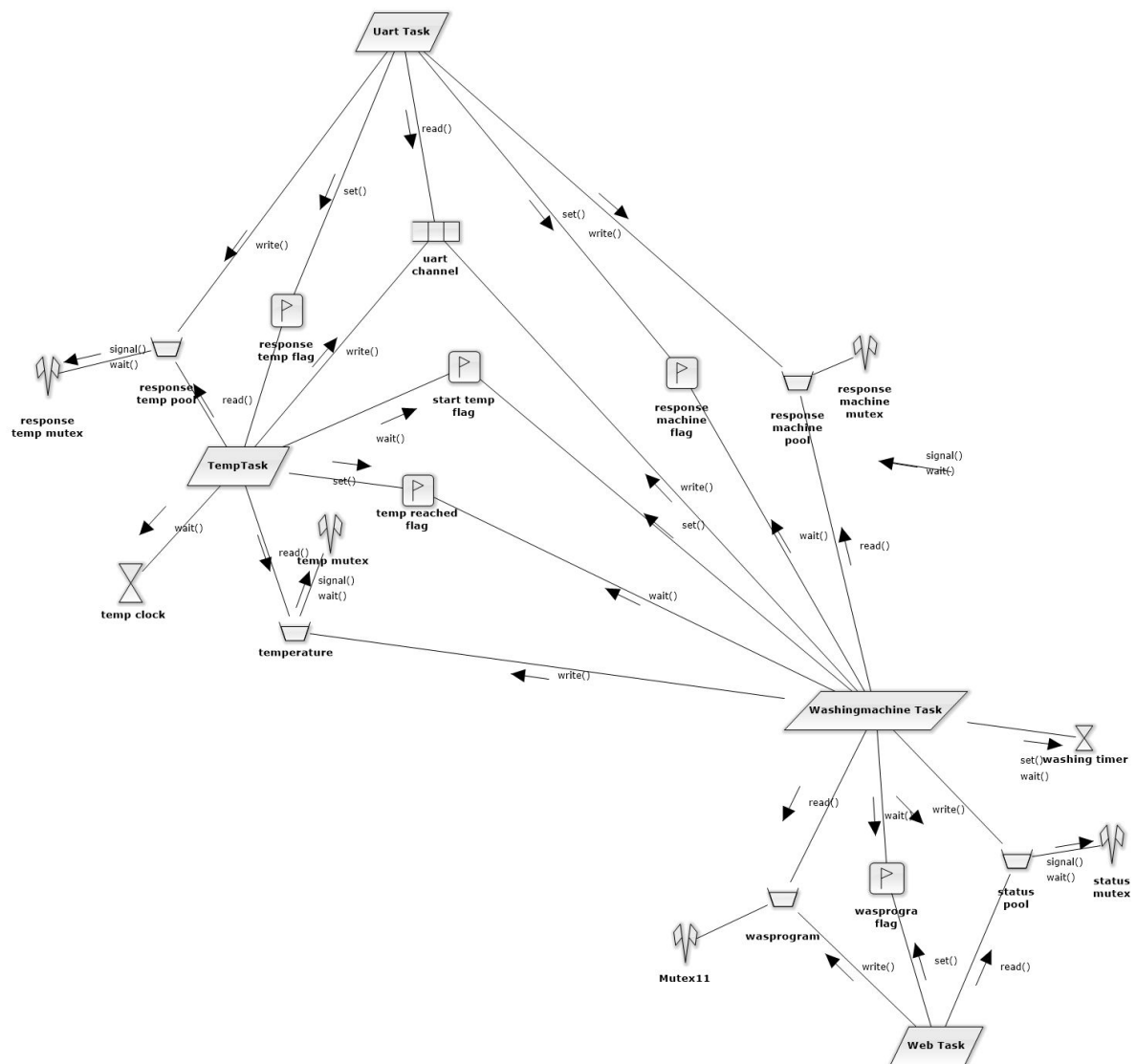


# VI Concurrency Diagram

## VI.I Hele diagram

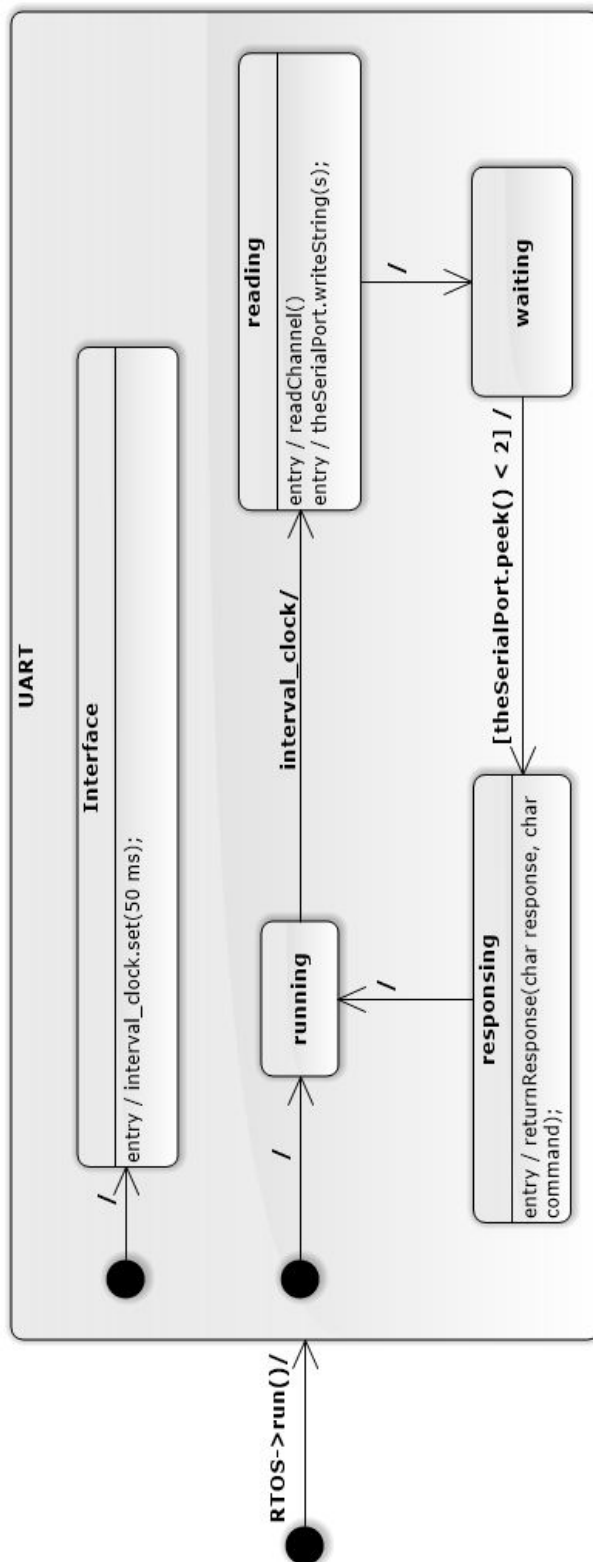


## VI.II Temperatuur deel

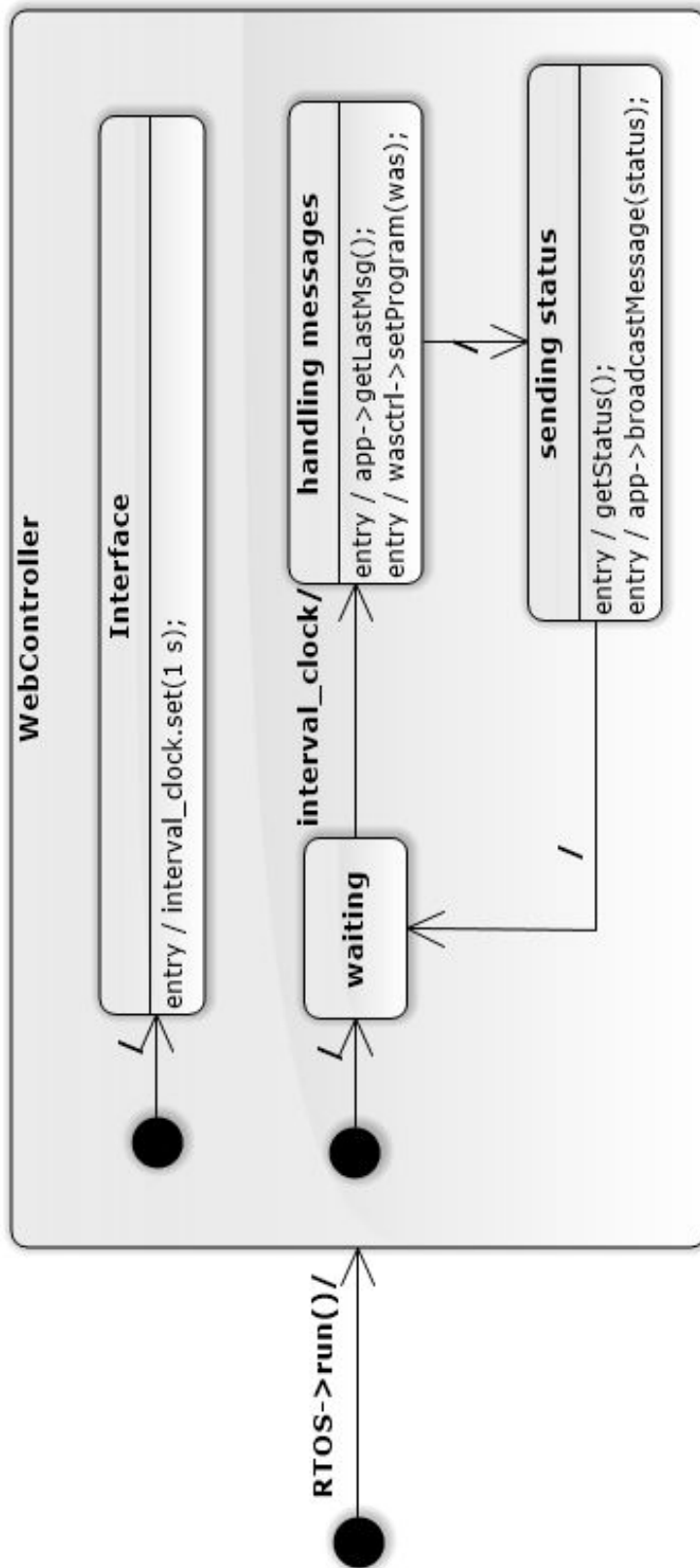


## VII State Transition Diagram

### VII.I UART

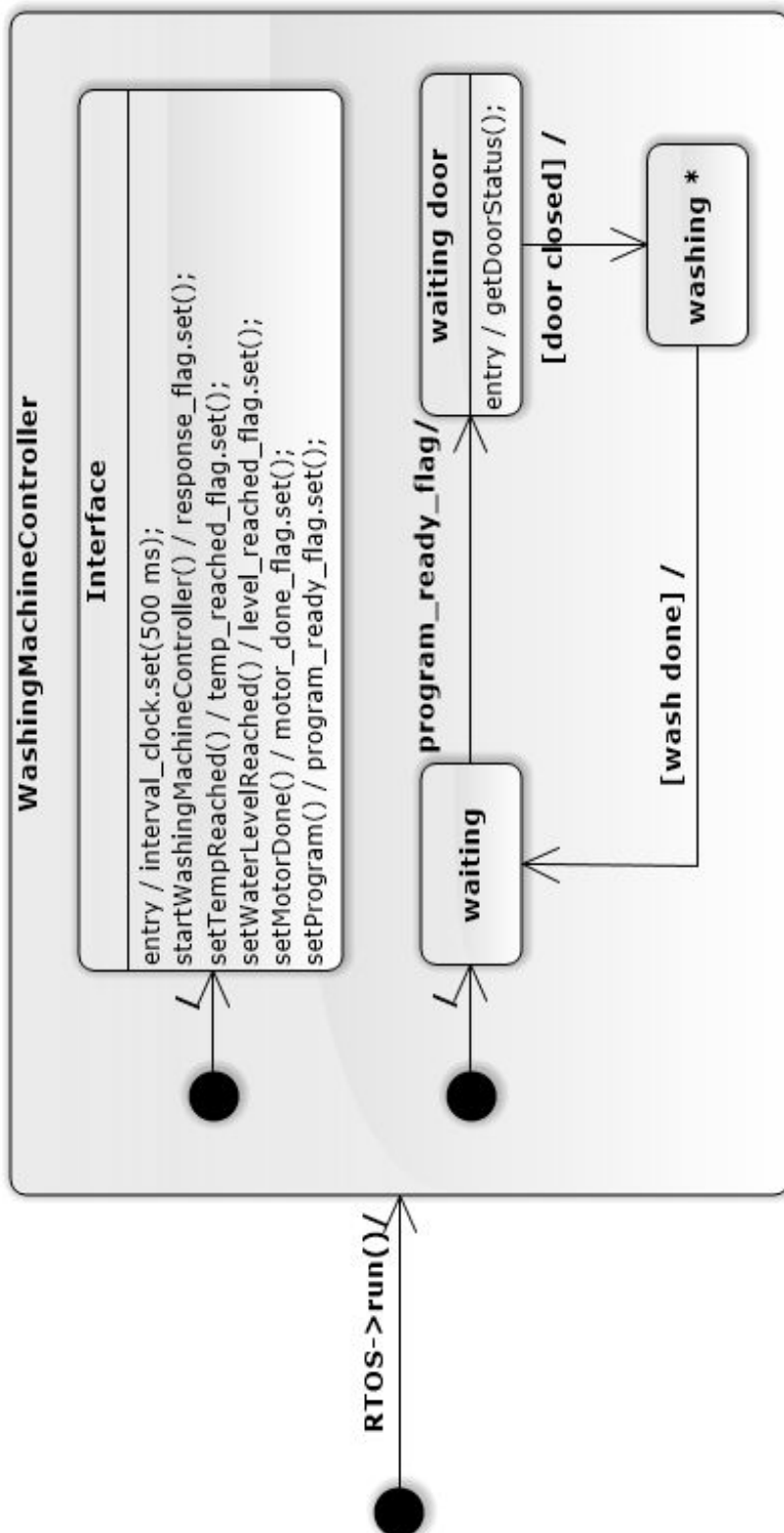


## VII.II WebController

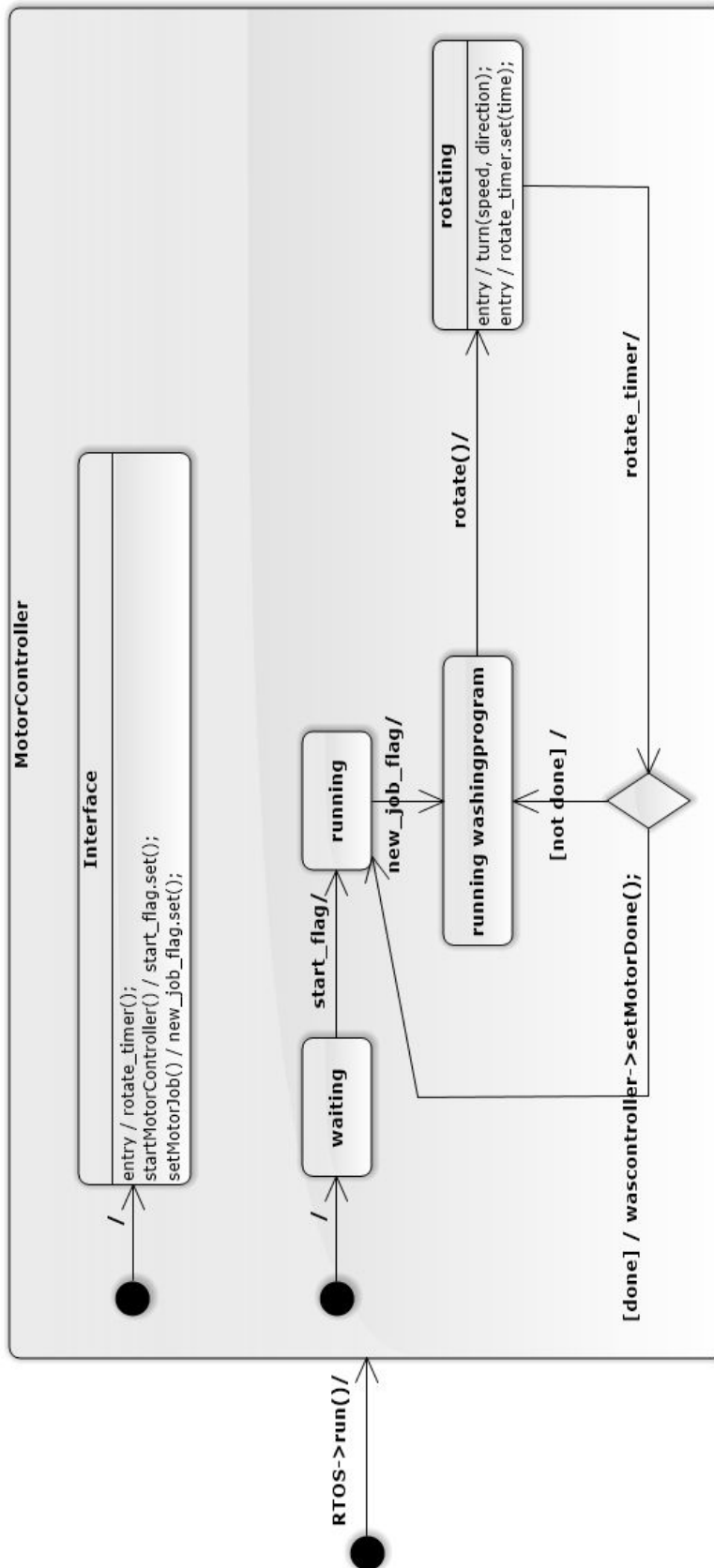




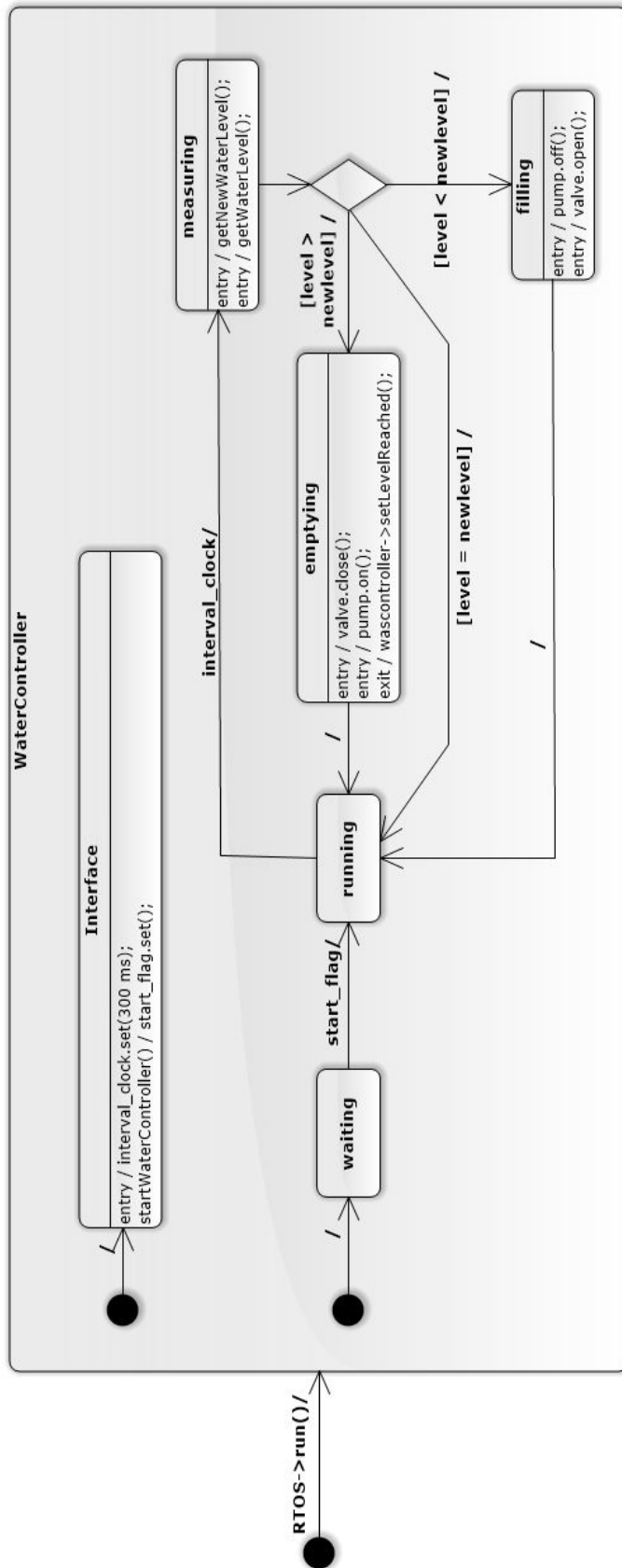
### VII.III WashingMachineController



## VII.IV MotorController



## VII.V WaterController



## VII.VI TempController

