Technisch verslag

Team: 4

Wouter van den Heuvel	1564952
Wilco Louwerse	1655993
~~Robin Noten~~	~~1671668~~
Daniel Klomp	1661521

Datum: 25-1-2016

Versie: 1.0

Managementsamenvatting

Hier volgt een korte samenvatting over het gehele verslag.
Op verzoek van de Hogeschool Utrecht heeft het team vanaf de grond af een wasmachine (emulator) gebouwd wat bediend kan worden vanaf een website. In dit verslag wordt beschreven hoe de wasmachine en de website zijn opgebouwd.

Alle informatie die is gebruikt bij het uitvoeren van de opdracht zijn verkregen door de cursussen C++ Programmeren & Software Engineering 2, Realtime System Programming, Operating Systems en Netwerk Programmeren en Themaopdracht 6 te volgen aan de Hogeschool Utrecht(2015-2016). In deze cursussen wordt aangeleerd hoe een systeem van de grond af kan worden ontworpen en ontwikkeld met C++ in combinatie met HTML en JavaScript.

Om de wasmachine en de bijbehorende website te ontwikkelen moeten eerst de Requirements worden opgesteld. In de Requirements worden veel belangrijke keuzes gemaakt. Een van die keuzes is dat de gebruiker in staat moet zijn om zijn eigen wasprogramma aan te maken en te kunnen veranderen. Elke gebruiker heeft dus een profiel met daarin de bijbehorende wasprogramma's.

Na het opstellen van de Requirements moet het systeem volledig worden ontworpen in de Solution Architecture. Ook bij de Solution Arcitecture zijn een aantal belangrijke keuzes gemaakt. een belangrijke keuze is bijvoorbeeld dat het systeem draait op vier lopende taken. Deze taken blijven constant met elkaar communiceren en wisselen gegevens uit.

Verder is het belangrijk te melden dat het eindproduct niet volledig is afgerond wegens het wegvallen van een teamgenoot. Hierdoor kwam het team in tijdsnood en kon het niet alle vooraf gestelde eisen in het systeem implementeren.

Uit de Requirements en de Solution Architecture kan worden geconcludeerd dat het systeem gebruiksvriendelijk is opgebouwd. Verder wordt geconcludeerd dat het systeem door middel van een duidelijk en snel takenstructuur zeer efficiënt met de gegeven data kan omgaan.

Aan een eventuele opvolger wordt geadviseerd veel te testen met de Raspberry Pi en de webserver, omdat door de tijdsnood het systeem nog niet volledig is getest.

Inhoud

1 Inleiding

Dit project heeft het team instaat gesteld om te leren programmeren in C++, een systeem vanaf de grond op te bouwen en om de vaardigheid van samenwerken met teamgenoten en de opdrachtgever onder de knie te krijgen.

In dit verslag wordt uitgelegd hoe de wasmachine tot stand is gekomen, hoe het functioneel en technisch in elkaar zit, en hoe een eventuele opvolger verder zou kunnen gaan met het ontwikkelen van de wasmachine. Dit gaat verslag verder in op een aantal belangrijke zaken die komen kijken bij dit project.

Eerst worden de **onderzoeken** die zijn verricht behandeld. Hierin worden alle onderzoeken en experimenten die zijn uitgevoerd tijdens het project nader toegelicht. Uit deze onderzoeken worden veel keuzes over het ontwerp van de wasmachine duidelijk

Verder wordt ingegaan op de **Requirements Architecture(**RA**)**, waarin een samenvatting wordt gegeven van de functionele en niet-functionele eisen aan de wasmachine behandeld. Hierin worden ook de use-cases het van systeem en de beschrijvingen hiervan nader uitgelegd.
Vervolgens wordt de **Solution Architecture(**SA**)** toegelicht. Hierin staat tot in de details uitgelegd hoe de wasmachine functioneert. In het SA wordt het klassendiagram, de taakstructurering en de State Transition Diagrams behandelt en worden een aantal ingewikkelde algoritmes toegelicht.
Tevens wordt **de realisatie** van de wasmachine behandeld. Een aantal belangrijke vragen worden beantwoord zoals: Wat heb ik nodig om zelf deze wasmachine te maken? Ook wordt uitgelegd hoe het ontwikkel proces eruit heeft gezien en welke problemen aan de orde zijn geweest en hoe die zijn opgelost.
Ook wordt het project nog **geëvalueerd**. Hier wordt besproken wat goed en wat minder goed ging tijdens het maken van de wasmachine en hoe dit de volgende keer beter kan gaan.

Ten slotte wordt de conclusie besproken, hierin worden de fictionele en technische aspecten van de wasmachine samengevat en wordt een conclusie getrokken. Tevens worden aanbevelingen gegeven aan een opvolgde programmeur, waar in wordt geadviseerd aan welke aspecten van het project verder te werken en op welke wijze dit te doen.

2. Onderzoek

Om het project goed te kunnen uitvoeren heeft het team een aantal onderzoeken moeten uitvoeren. Wat deze onderzoeken inhouden en wat de resultaten hier van zijn komt in dit hoofdstuk aan de orde.

2.1 Informatie verzameld over:

Het team heeft informatie verzameld over de werking van de Raspberry Pi. Vanuit de school is niet veel informatie vergeven over hoe de Pi werkt en hoe het beste omgegaan met Linux op de Pi.
Verder moest het team zelf uitzoeken hoe een website kan worden gebouwd met HTML, Css en Javascript. De informatie om een HTML website te bouwen is voornamelijk verkregen van http://w3schools.com . Op deze website wordt duidelijk uitgelegd hoe HTML, Css en Javascript werkt.

2.2 Experimenten die zijn uitgevoerd:

Een groot en vooral leerzaam experiment is vooral het laten werken van de Webserver. Vanuit de Hogeschool Utrecht is een webserver geleverd. In het experiment probeerde het team deze webserver om te bouwen om hem te laten werken met de Raspberry Pi. Helaas is dat niet gelukt en is toch een webserver van buitenaf gebruikt.
Verder is veel geëxperimenteerd met de lay-out van de webserver. Door goed te kijken naar andere grootte websites zoals Facebook een Google zijn veel ideeën tot stand gekomen en de meeste ideeën zijn daarna uitgewerkt tot een proefversie. Door veel proefversies te hebben kan het team de meest gebruiksvriendelijke lay-out uitkiezen.

3. Requirements Architecture

In de Requirements Architecture worden de functionele en niet-functionele eisen van het systeem behandel. De niet-functionele eisen worden als eerste behandeld in de vorm van de MoSCoW methode. Daarna komen de functionele eisen aan bod in de vorm van de Use-Case diagrammen.

3.1 Requirments:

In de MoSCoW methode (Zie bijlage 2) worden de niet functionele eisen aan het systeem vastgesteld. De MoSCoW bestaat uit Must, Should, Could en Won't Haves. In de Must Haves staan de eisen die zeker in het systeem moeten zitten. Verder staan in de Should Haves de eisen die, mits genoeg tijd, ook erg prettig zijn om in het systeem te hebben. De Could Haves zijn ideeën die waarschijnlijk niet in het systeem zullen zitten. En als laatste de Won't Haves, waarvan al zeker is dat die eisen niet in het systeem zullen worden geplaatst.

Hieronder staat een samenvatting van de belangrijkste eisen. In bijlage 2 staat de volledige MoSCoW-eisen uitgewerkt.

3.1.1 Must haves

- De gebruiker moet via de server verbinding kunnen maken met de wasmachine.
- De gebruiker moet een wasprogramma kunnen kiezen.
- De gebruiker moet een wasprogramma kunnen aanzetten/pauzeren/stoppen.
- De machine moet een wasprogramma kunnen draaien.

3.1.2 Should haves

- De gebruiker zou een wasprogramma moeten kunnen samenstellen en opslaan.
  - De gebruiker zou een eerder zelf gemaakt wasprogramma moeten kunnen wijzigen.
- De gebruiker zou een visuele weergave van het huidige wasprogramma en de status daarvan kunnen zien.
- De gebruiker zou de eindtijd en de tijdsduur moeten kunnen zien op het web interface.
- De gebruiker zou met een schuifbalk door het wasprogramma moeten kunnen lopen.

3.1.3 Could haves

- Om verbinding te maken met de server van de wasmachine kan de gebruiker een wachtwoord worden ingesteld.
- De gebruiker zou een tijdlijn kunnen zien waarop zichtbaar is, in welk stadium het wasprogramma zich bevind, wat al klaar is en wat er nog gaat gebeuren.

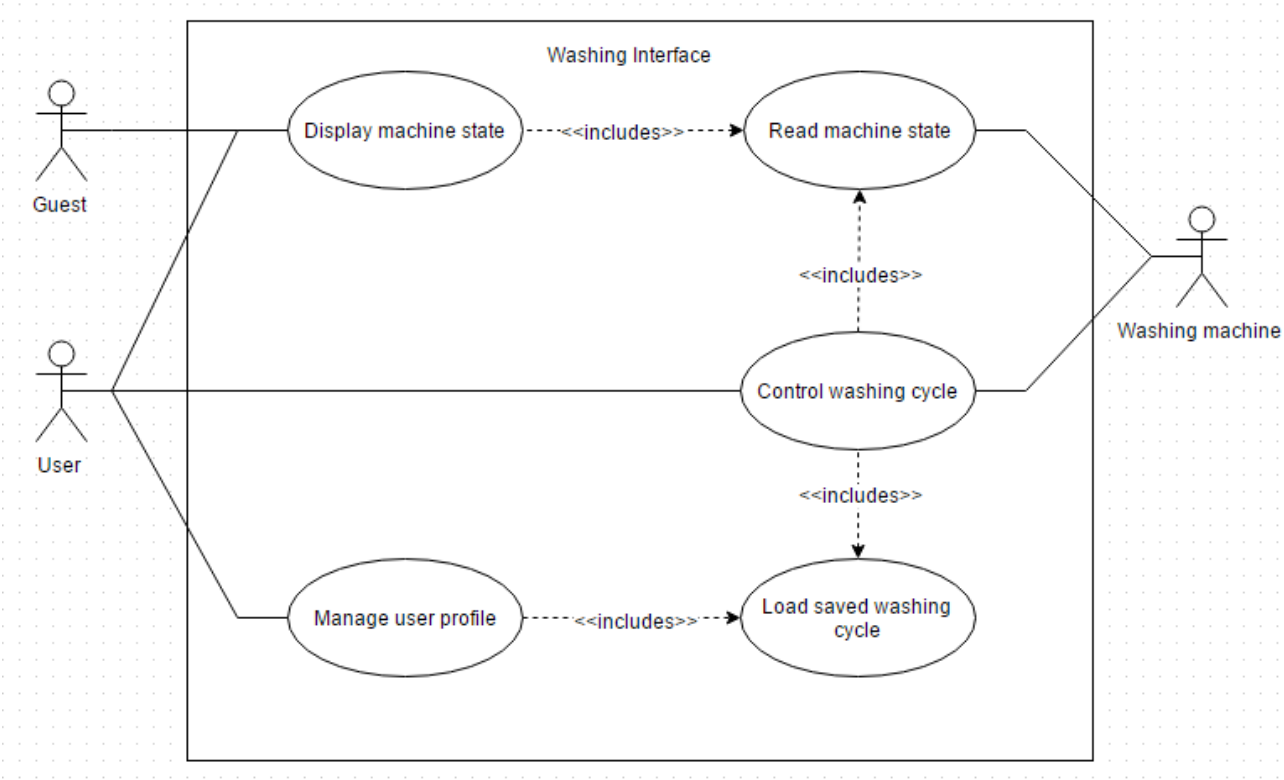
3.1.4 Will-not haves

- Er zal geen detectie zijn voor teveel was in de trommel.
- Er zal geen domeinnaam voor de webinterface geregistreerd worden.
- Er zal geen functionaliteit worden geïmplementeerd om vanuit één webinterface meerdere wasmachines aan te sturen.

3.2 Use case diagram

Het *use case model* geeft gemakkelijk en overzichtelijk weer wat de functionele eisen zijn van de wasmachine.

Het systeem moet de gebruiker instaat stellen om het wasprogramma te controleren. Om dat te kunnen doen, moet een wasprogramma geladen worden uit het profiel van de gebruiker en moet de status van de machine worden opgevraagd.
Verder moet de gebruiker instaat worden gesteld om zijn profiel aan te kunnen passen. In dit profiel bevinden zich het wachtwoord en alle wasprogramma van de gebruiker. Hier kan de gebruiker dus ook nieuwe wasprogramma's aanmaken of al bestaande wasprogramma's aanpassen.



3.4 Use case Beschrijvingen

In de use-case beschrijvingen worden de use-cases verder uitgelegd. Elke use-case heeft een doel, wat de functie van de use-case beschrijft.

Use case naam	**Read machine state**
Doel	Het uitlezen van de staat van de aangesloten wasmachine.
Pre-condities	geen
Post-condities	De staat van de wasmachine is uitgelezen.
Uitzonderingen	Er is geen wasmachine aangesloten.

Use case naam	**Control washing cycle**
Doel	Actor ‘User’ in staat stellen om de wasmachine aan te sturen via het web interface
Pre-condities	De wasmachine is aangesloten en heeft connectie met het web interface
Post-condities	De wasmachine heeft het wasprogramma uitgevoerd
Uitzonderingen	De Actor ‘User’ kan zijn eigen wasprogramma’s eerst wijzigen voordat het wasprogramma draait

Use case naam	**Load stored washing program**
Doel	Een eerder opgeslagen programma wordt uit opslag voor lange termijn geladen
Pre-condities	Een gebruiker heeft connectie met de wasmachine en is ingelogd
Post-condities	Het programma is gereed voor gebruik en/of weergave
Uitzonderingen	Geen

Use case naam:	**Manage user profile**
Doel	Beheert alle gebruiker profielen, slaat nieuwe informatie van nieuwe of oude gewijzigde profielen op en verleent de informatie van deze profielen wanneer daar om gevraagd wordt.
Pre-condities	Er wordt nieuwe informatie van een profiel gestuurd of er wordt gevraagd om al bestaande informatie van een profiel.
Post-condities	De nieuwe profiel informatie wordt verwerkt of de gevraagde informatie van een bestaand profiel wordt terug gegeven.
Uitzonderingen	Wanneer er wordt geprobeerd een profiel aan te maken met een al bestaande gebruikersnaam wordt dit nieuwe profiel niet aangemaakt.

Use case naam:	**Display Machine State**
Doel	De huidige staat van de machine (waterniveau, temperatuur etc.) weergeven via de webbrowser.
Pre-condities	De machine staat aan
Post-condities	geen
Uitzonderingen	Als een gebruiker nog niet is ingelogd, wordt de gebruiker verwezen naar de inlogpagina.

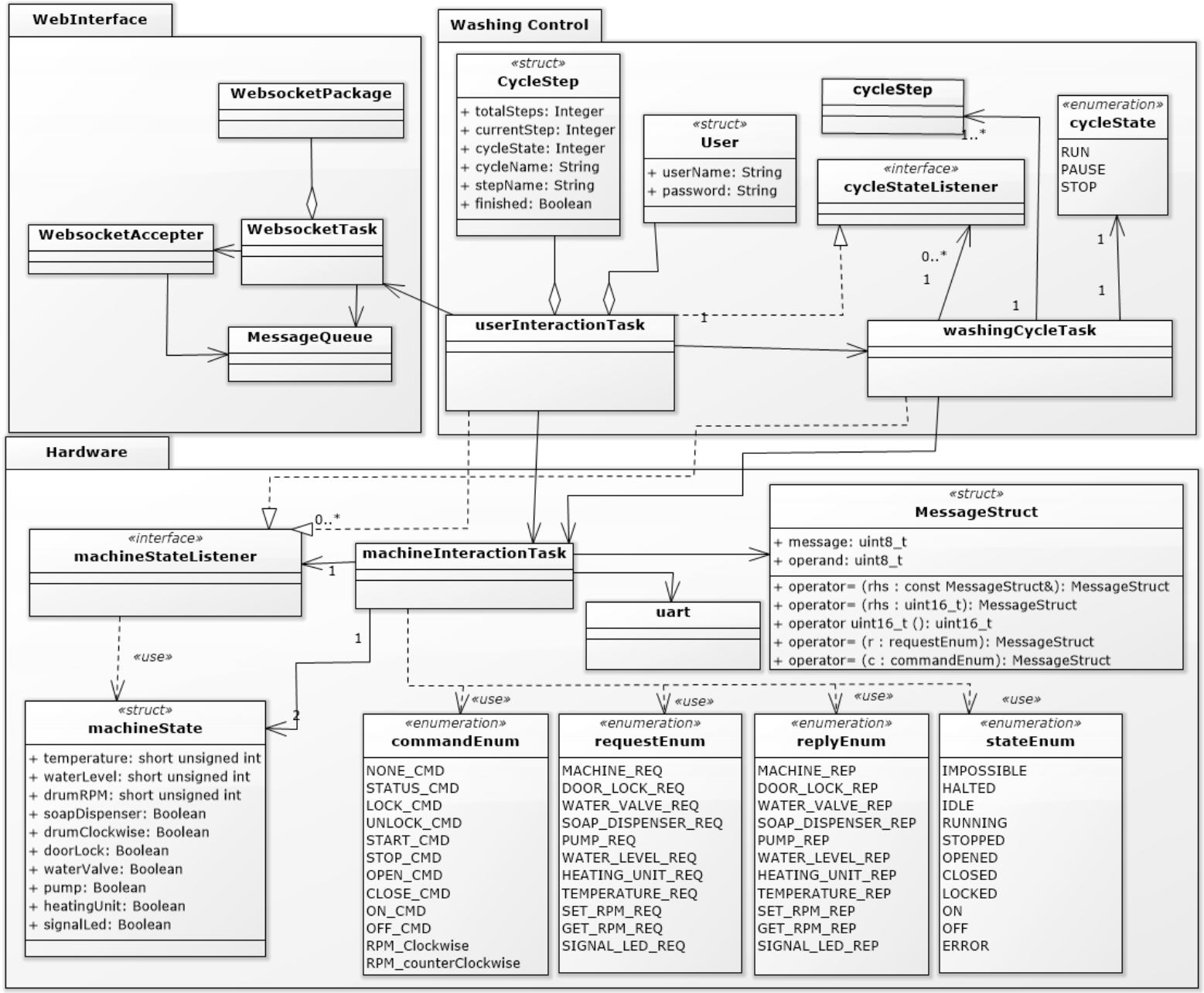
3.5 Activity Diagrammen

In de Activity Diagrammen worden stap voor stap uitgelegd hoe systeem in elkaar zit. In deze diagrammen wordt vanaf het opstarten behandeld hoe het systeem functioneert. Uit deze diagrammen wordt dus duidelijk hoe het aanmaken van een wasprogramma werkt, hoe een wasprogramma draait en hoe de gebruiker door de webbrowser interface profiel instellingen kan wijzigen.
Deze diagrammen staan allemaal in de bijlagen(Bijlage 9.3 *“Requirements Actitecture”*).

4. Solution architecture

4.1 Klassen Diagram

(zie bijlage 9.4 “Volledig Klassen Diagram” voor een uitgebreidere versie van dit diagram)



Het klassendiagram brengt in beeld wat de connecties zijn tussen klassen binnen het systeem. Door middel van dit diagram wordt ook duidelijk hoe en waarom bepaalde klassen met elkaar communiceren. Voor dit project staat het klassendiagram op de vorige pagina. Dit is niet het gehele klassendiagram, omdat de methodes en attributen van de klassen zijn weggelaten. Het volledige uitgewerkte klassendiagram staat in de bijlagen(Hoofdstuk 9,4).

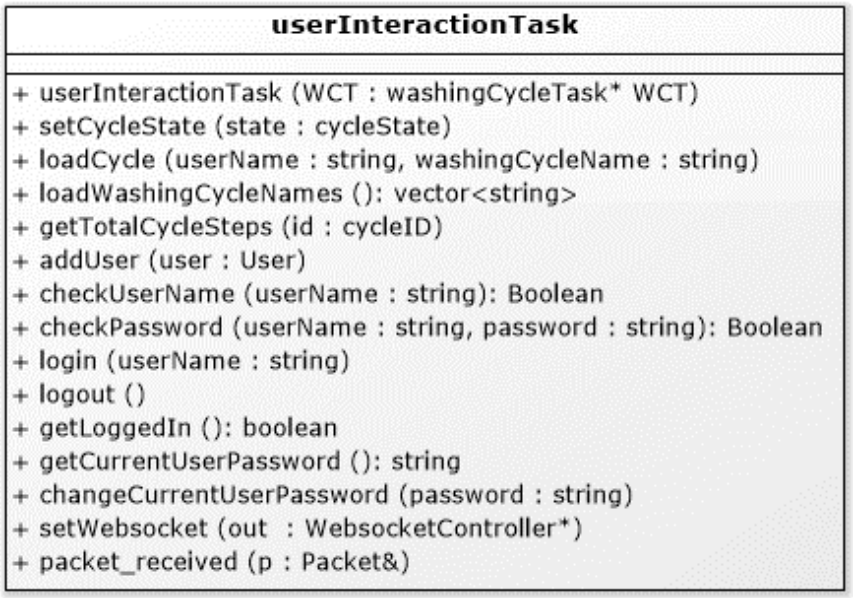
Voor zowel de washingCycleTask als de machineInteractionTask is er voor gekozen om het Listener Pattern te gebruiken, zodat andere klassen relatief eenvoudig statusberichten kunnen ontvangen met betrekking tot de voortgang van een wasprogramma, of de huidige status van het systeem.

De keuze om de machineInteractionTask verantwoordelijk te maken voor het beheren van de temperatuur en waterniveau komt voort uit het feit dat deze klasse de statistieken nauwlettend moet observeren. Ook zorgt dit ervoor dat de kennis van het fysieke systeem op één enkele plaats bestaat.

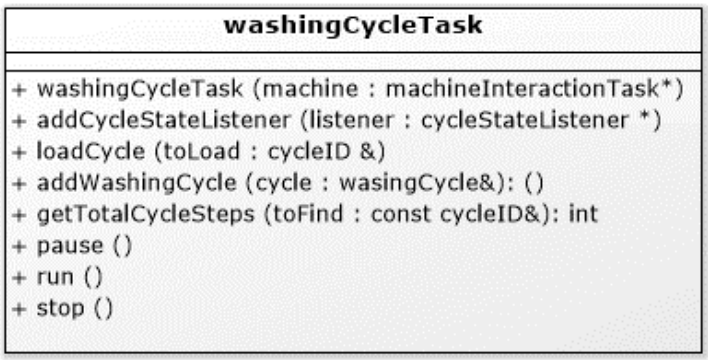
In plaats van aparte Boundary klassen voor alle onderdelen van de wasmachine is er voor gekozen om deze details te verbergen in de MachineInteractionTask, allereerst om de complexiteit van klassen die bij hardware details moeten kunnen komen te beperken, en omdat de hardware is afgeschermd door middel van de UART.

UserInteractionTask:
De UserInteractiontask is verantwoordelijk voor het communiceren tussen de WebsocketTask en de WashingCycleTask. De taak leest uit de CycleStateListener en de MachineStateListener de status van het wasprogramma en de status van de wasmachine om die weer te kunnen geven op de website.

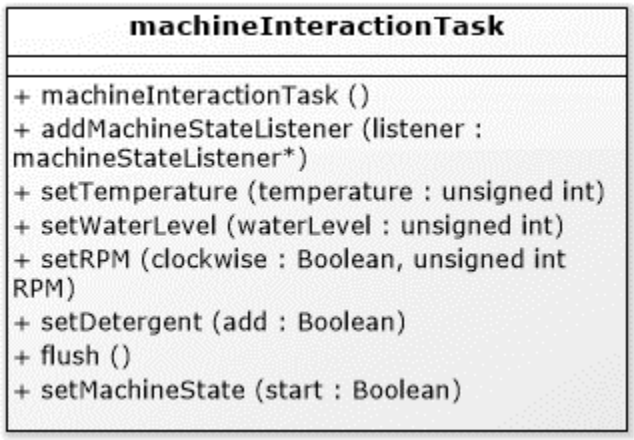
Verder stuurt de UserInteraction Task berichten door vanaf de WebsocketTask naar bijvoorbeeld de WashingCycleTask. Voorbeeld: als de gebruiker het wasprogramma wilt pauzeren, dan stuurt de WebsocketTask dat bericht naar de UserInteractionTask, zodat de taak dit vervolgens door kan sturen naar de washingCycleTask.



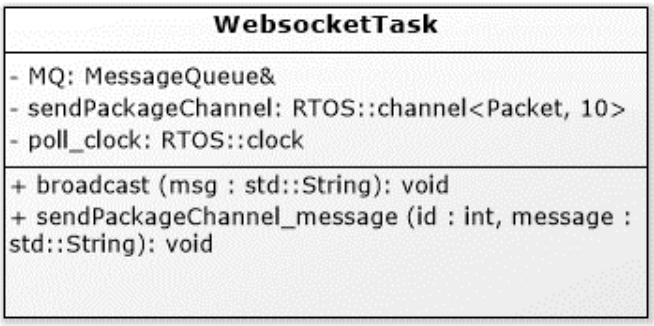
WashingCycleTask:
De WashingCycleTask is verantwoordelijk voor het draaien van het wasprogramma. De taak communiceert met de wasmachine via de MachineInteractionTask om het wasprogramma uit te voeren. Ook stuurt de washingCycleTask de status van het wasprogramma naar alle CycleStateListeners, zodat andere taken kunnen weten hoever het wasprogramma is.



MachineInteractionTask:
De MachineInteractionTask is verantwoordelijk voor de communicatie tussen de wasmachine en de washingCycleTask. Dit doet hij door middel van de Uart klassen. Ook stuurt de MachineInteractionTask de status van de wasmachine (temperatuur, waterlevel, etc.) naar alle MachineStateListeners zodat deze taken weten in welke status de wasmachine verkeerd.



WebsocketTask:
De WebsocketTask is verantwoordelijk voor de communicatie tussen de Websocket en de UserInteractionTask. De WebsocketTask krijgt van de UserInteractionTask de status van de wasmachine en van het huidig draaiende wasprogramma. Dit wordt vervolgens doorgestuurd naar de Websocket. Daarnaast kan de WebsocketTask de commando's RUN, PAUSE en STOP krijgen van de Websocket als deze commando's worden gegeven vanaf de website.



4.2 Taakstructurering

De taakstructurering is gemaakt als vervolg op het klassendiagram.
In dit diagram zijn bepaalde klassen samengevoegd tot groepjes om nog meer duidelijkheid te krijgen van de communicatie binnen het systeem en om te voorkomen dat meerdere gelijksoortige taken vergelijkbare dingen gaan doen.

4.2.1 Objecten

Object	ID	Object omschrijving	Type taak	Periode	Deadline	Prioriteit
WashingMachine	1	Interface voor fysieke wasmachine	Asynchroon I/O	-	250ms	0
WebSocket	2	Internet interface	Asynchroon I/O	-	500ms	2
UART	3	Comminucatie tussen de wasmachine en het Rtos	Asynchroon I/O	-	50ms	0
DisplayController	4	Use case “Display machine state” Bepaalt wat wordt laten zien op het display	Asynchroon	-	500ms	2
MachineRead Controller	5	Use case “Read machine state” Laat de WashingMachine periodiek de status updaten.	Periodiek	500ms	250ms	0
WashingCycle Controller	6	Use case “Control washing cycle”	Periodiek	500ms	250ms	1
UserProfile Controller	7	Use case “Manage user profile”	Asynchroon	-	500 ms	3
LoadCycle Controller	8	Use case “Load saved washing cycle” Laadt de WashingCycle en slaat dit ook op.	Asynchroon	-	500 ms	3

In de opsomming van de objecten op de vorige pagina zijn alle losstaande Boundary objecten van de wasmachine samengevoegd tot een object, de Washingmachine. In dit object zitten alle bijbehorende objecten van de wasmachine, zoals de deurvergrendeling, de motor en de noodknop. Een compleet overzicht van de objecten staan vermeld in de wasmachine emulator beschrijving(Wensink, M. *“Beschrijving wasmachine-emulator”*).

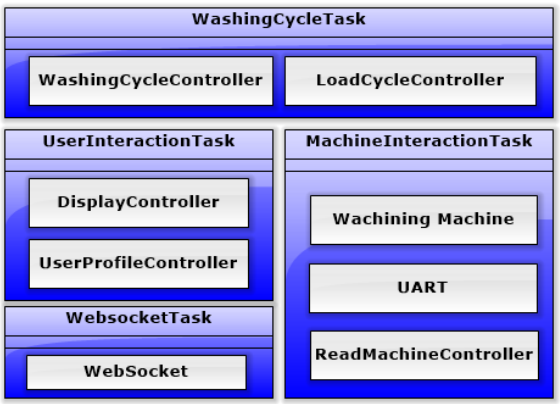
4.2.2 Taken

Om de objecten goed met elkaar te kunnen laten communiceren, en om het systeem snel en efficiënt te maken, worden de objecten samengevoegd tot taken. Deze taken lopen(“Runnen”) constant in het systeem.

De MachineInteractionTask bestaat uit de Washing Machine, de UART en de ReadMachineController. Deze taak heeft de verantwoordelijkheid om met de objecten van de wasmachine te communiceren. Dat houdt in berichten versturen en de status van de wasmachine ontvangen. Deze taak wordt om de 500ms aangeroepen (periodiek) om de status van de wasmachine op te vragen en nieuwe berichten naar de wasmachine te versturen. Dit is 500ms omdat de status van de wasmachine niet drastisch zal veranderen in een kortere periode dan 500ms, maar langer dan 500ms is te lang. De deadline voor deze taak is 250ms omdat de taak snel uitgevoerd moet worden, echter, de UART heeft een vertraging van 10ms per bericht, wat de taak zal ophouden.
Deze taak heeft de hoogste prioriteit omdat een bericht naar de wasmachine, bijvoorbeeld een noodgeval om te stoppen, altijd voor gaat.

De WashingCycleTask bestaat uit de washingCycleController. Deze taak heeft de verantwoordelijkheid om de veranderingen van de wasmachine te analyseren en te vergelijken met het huidige wasprogramma. Verder wordt deze taak periodiek aangeroepen om de 500ms omdat de MachineInteractionTask om de 500ms een nieuwe status levert aan de WachingCycleTask. De deadline voor deze taak is 250ms omdat het systeem niet teveel vertraagd, maar wel druk zet achter de taak.
Deze taak heeft een prioriteit lager dan de MachineInteractionTask omdat de washingCycleTask wel snel berichten moet versturen naar de MachineInteractionTask, maar de berichten naar de wasmachine zelf belangrijker zijn.

De UserInteractionTask is verantwoordelijk voor het communiceren tussen het RTOS en de WebsocketTask. De UserInteractionTask ontvangt de berichten van de WebsocketTask en stuurt de status van de wasmachine (MachineInteractionTask) door naar de WebsocketTask. Deze taak wordt asynchrone aangeroepen door de WebsocketTask of de MachineInteractionTask. De deadline voor deze taak is 500ms, omdat het doorsturen en ontvangen van berichten van de WebsocketTask niet erg tijdsgebonden is.

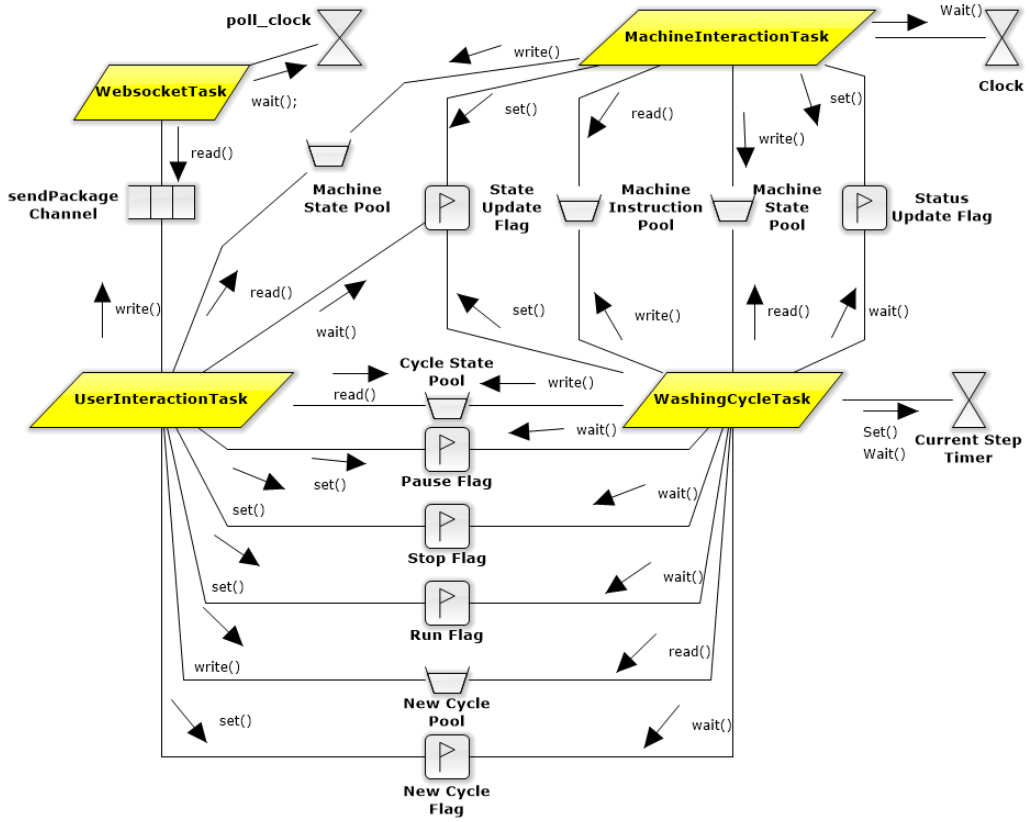


De WebsocketTask is verantwoordelijk voor het ontvangen en doorsturen van informatie tussen het RTOS en de Websocket. De WebsocketTask ontvangt de berichten van de UserInteractionTask en stuurt de ontvangen status van de wasmachine door naar de Websocket. Deze taak wordt asynchrone aangeroepen door de Websocket of de UserInteractionTask. De deadline voor deze taak is 500ms, omdat het doorsturen en ontvangen van berichten van de Websocket niet erg tijdsgebonden is.

Taak	Onderdelen	Type taak	Periode	Deadline	Prioriteit
MachineInteractionTask	1,3,5	Periodiek	500ms	250ms	1
WashingCycleTask	6, 8	Periodiek	500ms	250ms	2
UserInteractionTask	4,7	Asynchrone I/O-taak	-	500ms	3
WebsocketTask	2	Asynchrone I/O-taak	-	500ms	3

4.3 Concurrency Diagram

In het Concurrency diagram wordt de interactie tussen de verschillende taken duidelijk gemaakt. De taken binnen het systeem communiceren met elkaar via synchronisatie methodes.



In het Concurrency diagram voor de wasmachine praten/communiceren de meeste taken via Pools. Dit is omdat op de informatie die wordt doorgestuurd gewacht moet kunnen worden en dit de meest effectieve manier is om gegevens door te geven.

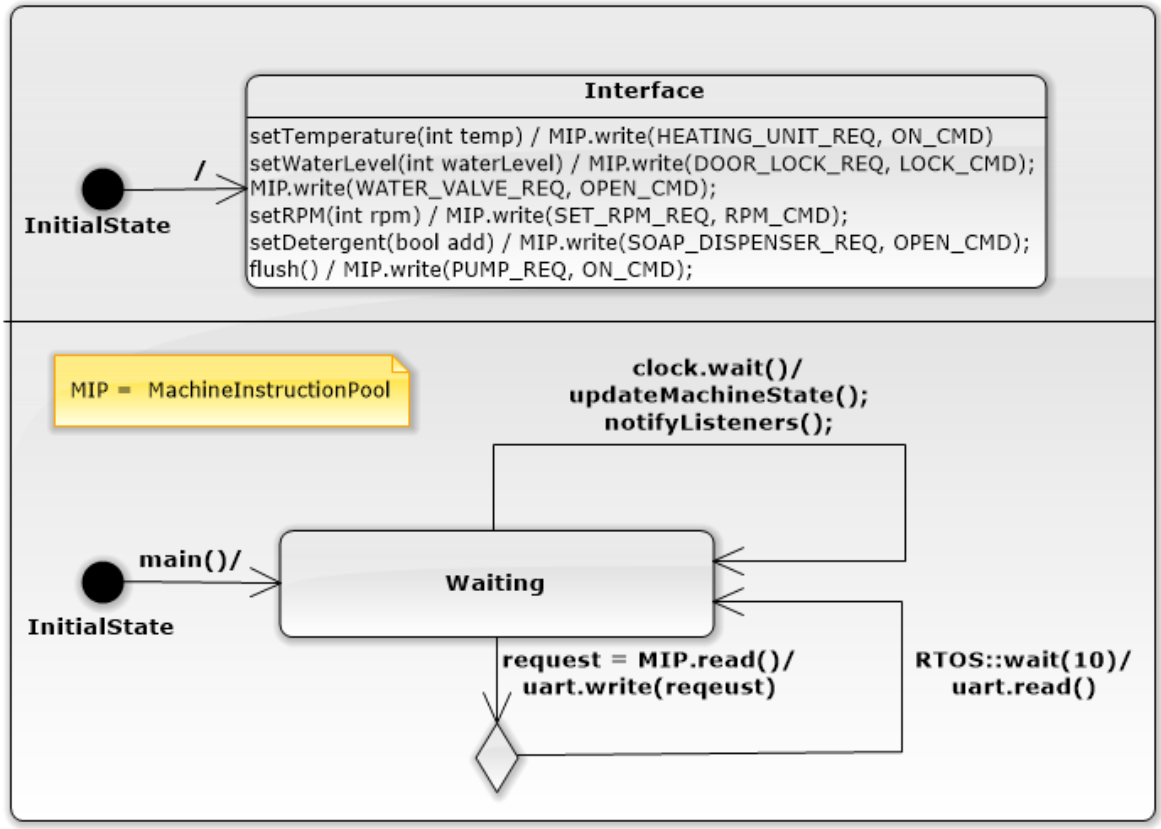
Een goed voorbeeld is de communicatie vanaf de MachineInteractionTask naar de UserInteractionTask en de WashingCycleTask door middel van de MachineStatePool. In deze communicatielijn schrijft de MachineInteractionTask de status van de wasmachine in de MachineStatePool. Dit had ook een Channel kunnen zijn, een mechanisme dat vrijwel het zelfde effect heeft als een Pool met een Flag. Het gebruik van een Pool met een Flag is echter handiger omdat daarbij altijd maar een (actuele) status word doorgegeven en wanneer er een nieuwe status word doorgegeven die de oude overschrijft. Zodat niet zoals het bij een Channel werkt oude statussen nog worden verwerkt terwijl je eigenlijk alleen de allerlaatst gestuurde status wilt gebruiken.

4.4 State Transition Diagrams

In de State Transition Diagrams(STD) wordt de volledige werking van het systeem uitgewerkt. Elke taak die het systeem uitvoert wordt volledig uitgewerkt. Verder is ook te zien hoe de taken met elkaar communiceren.

4.4.1 MachineInteractionTask

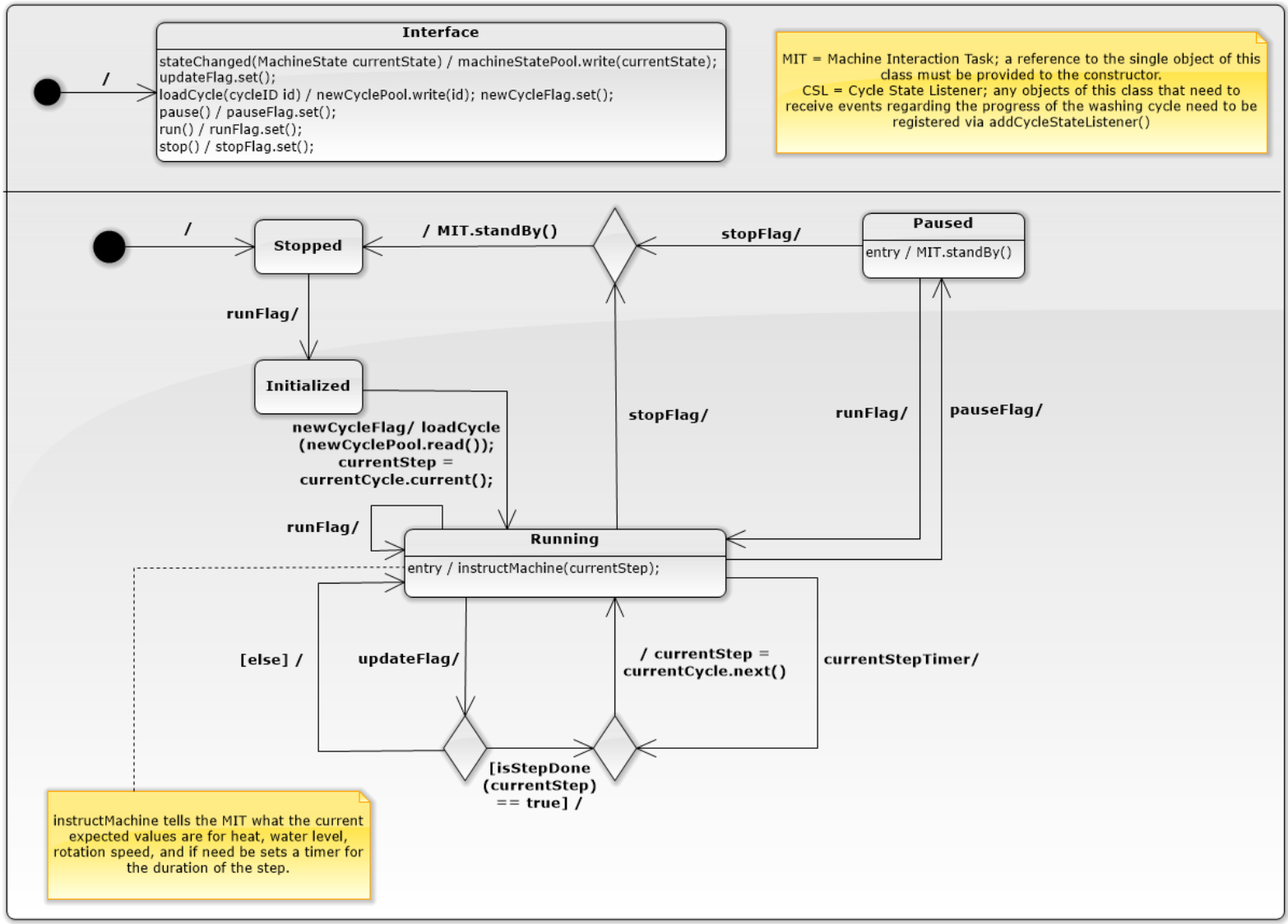
In de machineInteractionTask wordt alle communicatie met de tastbare wasmachine geregeld door bytes naar de Uart te sturen en de byte die terug wordt gegeven uit te lezen.



Bij het starten van het programma wacht de machineInteractionTask eerst totdat er ofwel een klokslag plaatsvindt, of tot er een stuuropdracht in de daarvoor toegewezen pool wordt gezet. Bij een klokslag worden de verschillende onderdelen van de wasmachine ondervraagt, en vind er een event plaats voor alle aangemelde listeners met de meest recente informatie. Als er een stuuropdracht in de pool staat wordt deze aan de machine doorgegeven, waarna de MIT eerst de response uitleest, voordat er wordt teruggekeerd naar de wachttoestand.

4.4.2 WashingCycleTask

In de washingCycleTask wordt het huidige wasprogramma uitgevoerd/bijgehouden. De status van het huidige wasprogramma kan van buiten af aangepast worden (pauzeren of stoppen), dit wordt ook in deze taak verwerkt. Naast dit stuurt de washingCycleTask ook naar alle geregistreerde cycleStateListeners wat de status is van het huidige wasprogramma en de fase waarin deze verkeerd.

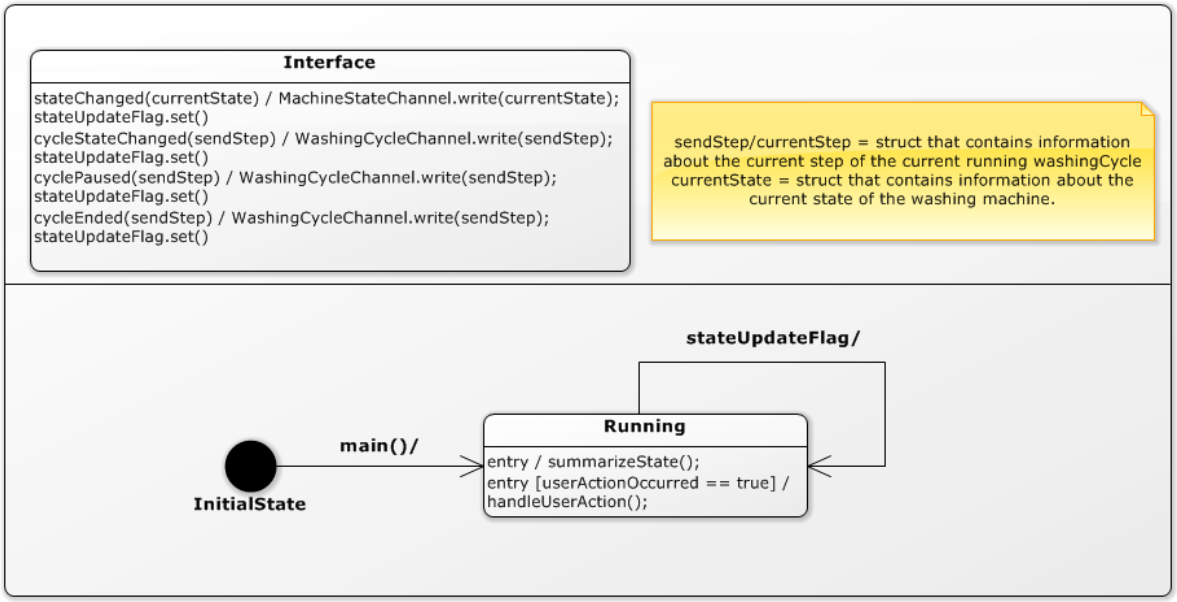


Bij het starten van het programma komt de washingCycleTask eerst in de ‘Stopped’ status terecht. In deze status wordt gewacht tot dat de run vlag gezet wordt, zodra dat gebeurd gaat deze taak door naar zijn ‘Initialized’ state. Zodra de washingCycleTask van zijn ‘Stopped’ status naar ‘Initialized’ status gaat wordt de loadCyclePool uitgelezen, hierin wordt gezet welk wasprogramma wordt gedraaid. Het uitgelezen wasprogramma wordt opgeslagen in de “ongoing” ‘washingCycle’ zodat de taak onthoudt welk wasprogramma hij aan het draaien is.

Na het vaststellen van het wasprogramma die moet draaien komt de washingCycleTask in een nieuwe staat (‘Running’)terecht. In deze grote loop wordt gekeken of de status van het wasprogramma moet veranderen van “RUN” naar “PAUSE” of “STOP”. Als dit niet het geval is blijf de status van deze taak ‘Running’ en zal hij alle fases van het wasprogramma een voor een uitvoeren. Wordt het wasprogramma gestopt of is het klaar, dan zal de status weer terug gaan naar ‘Stopped’. Ook is er nog een ‘Paused’ status, dit is ook een loop waarin de taak blijft vanaf wanneer de status van het wasprogramma “PAUSE” wordt tot dat het weer veranderd naar “RUN” of “STOP”. Waarbij in het geval van “RUN” de taak weer verder gaat in zijn ‘Running’ status en bij het geval van “STOP” terug komt in zijn ‘Stopped’ status.

4.4.3 UserInteractionTask

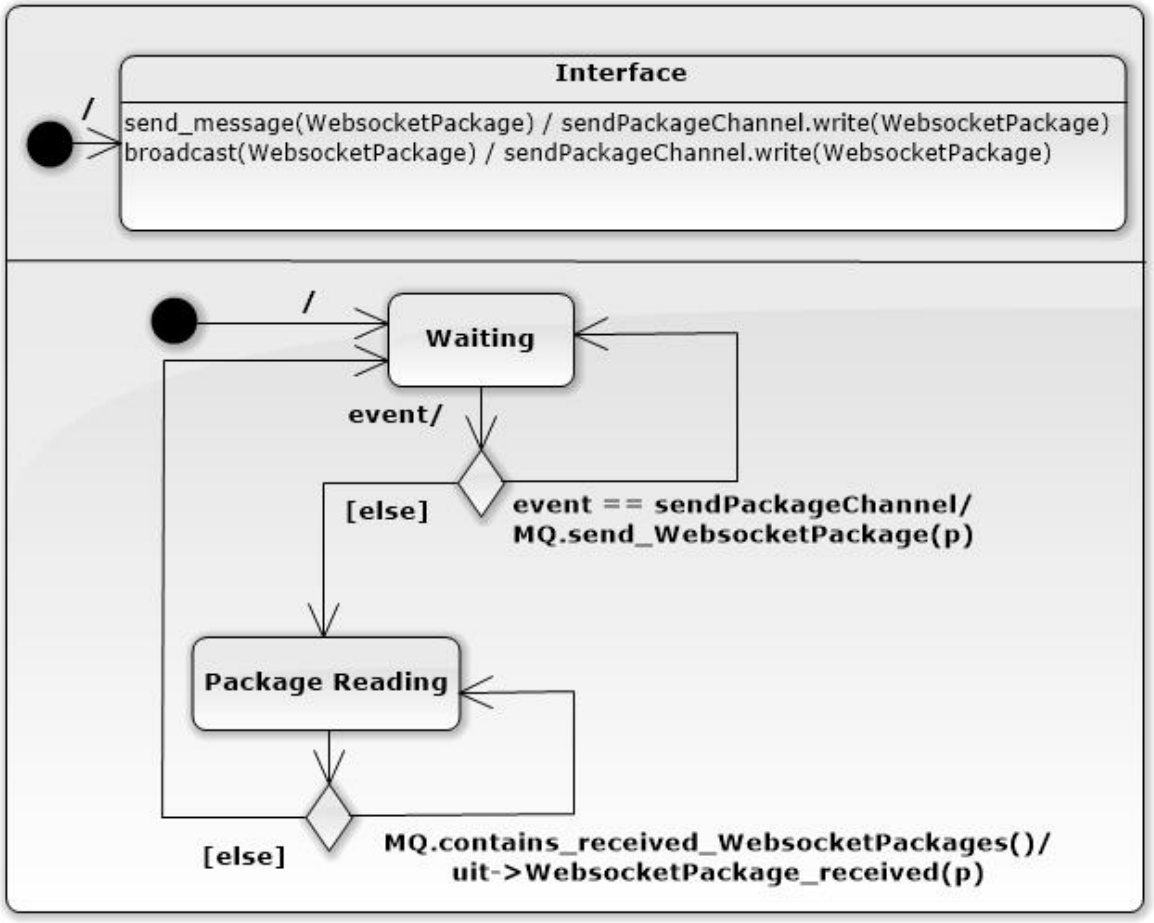
In de UserInteractionTask worden alle berichten die zijn verzonden door de gebruiker verwerkt en wordt de bijbehorende actie uitgevoerd.



Deze taak is constant aan het wachten totdat de stateUpdateFlag wordt gezet, dit gebeurd wanneer er van buiten deze taak een opdracht wordt gestuurd naar deze taak. Als dit gebeurd en de vlag wordt gezet dan zal deze taak de gegeven opdracht uitvoeren.
De opdrachten die gestuurd worden naar deze taak zijn opdrachten voor de Websocket en worden dus daar naar toe gestuurd, maar deze taak luistert ook naar de Websocket door middel van handleUserAction(). De uitgelezen opdrachten vanaf de Websocket worden doorgestuurd naar hun betreffende taak die deze opdracht moet uitvoeren.

4.4.4 WebsocketTask

In de WebsocketTask worden alle berichten die zijn verzonden door de gebruiker ontvangen en doorgestuurd naar de UIT (UserInteractionTask)



Deze taak is constant aan het wachten totdat er een event plaats vind, als dit event veroorzaakt is doordat er wat is geschreven in de sendPackageChannel (dit gebeurd wanneer er van buiten deze taak een opdracht wordt gestuurd naar deze taak) dan stuurt hij het verstuurde WebsocketPackage door naar de MQ (MessageQueue). Als het event dat plaats vond niet veroorzaakt is door de sendPackageChannel (maar door de klok) dan komt deze taak in een nieuwe status ‘Package Reading’, in deze status worden alle geschreven packages doorgestuurd naar de UIT.

5 Realisatie

In de realisatie worden de problemen en oplossingen die zich hebben voorgedaan in het project besproken. Ook worden de belangrijkste delen van de code behandeld en uitgelegd, zodat het grootste deel van de code duidelijk is.

5.1 Problemen

Bij het realisatie van het project is het team tegen een heel aantal problemen aangelopen. De belangrijkste problemen worden hier besproken.
Het eerste belangrijke probleem was dat het team halverwege het project van vier naar drie man werd verkleind. Een van de teamleden moest helaas wegens persoonlijke reden het team verlaten. Hierdoor kwam het team in grote tijdnood.
Dit probleem is opgelost doordat de overgebleven teamleden extra werkuren in het project hebben gestopt, en door een aantal eisen niet te realiseren. Zo kan een gebruiker niet een eigen wasprogramma aanmaken en werkt het aanmaken van een profiel niet.

Verder ging het maken van de code en het maken van de Emulator ook niet even soepel. Het team had het probleem dat de Raspberry Pi niet kon communiceren over de SSH. Daarnaast was het was het erg lastig om de Raspberry Pi een static Ip-adres te geven. Dit heeft de snelheid uit het project gehaald, doordat de code niet getest kon of gecompileerd kon worden. Dit is uiteindelijk opgelost, maar door het grote tijdsgebrek kon de code niet meer worden getest op de Raspberry Pi.

Het laatste grote probleem was met de webserver. Het was voor het team niet goed duidelijk dat de webserver niet zelf gemaakt hoeft te worden. De oplossing hiervoor was dus een webserver van buitenaf. Dit is op het laatste moment opgelost, maar ondertussen is veel tijd verloren gegaan aan het maken van een eigen webserver.

5.2 Algoritmen

Uart:
In de klassen ‘uart’ zijn er twee functies die gebruikt worden voor communicatie tussen de wasmachine en het systeem (MachineInteractionTask). Deze twee functies zijn de write() en read() functies. In de write() functie worden twee meegegeven bytes verstuurd naar de wasmachine door middel van de Libserial functie write(). De eerste byte die wordt meegestuurd is het request byte. Dit byte geeft aan naar welk hardware component een opdracht wordt gestuurd. Het tweede byte, het command byte geeft aan welk commando moet worden uitgevoerd op dit hardware component. Na het aanroepen van een write() via de ‘uart’ wordt er altijd een byte teruggeven die door middel van de read() functie wordt uitgelezen. Deze functie maakt gebruik van de libserial functie read(). Deze functie leest het ene byte uit dat terug gestuurd wordt, deze byte bevat de status van het hardware component waar naartoe geschreven werd.

6 Evaluatie

In de evaluatie wordt besproken wat allemaal goed ging en wat nog beter had gekund. Ook wordt uitgelegd wat dan anders had moeten gebeuren.

Om positief te beginnen behandelen we eerst de aspecten die goed gingen in het ontwikkelproces van de wasmachine.
In het begin van het ontwikkelproces ging de communicatie nog niet zo goed, maar dat is later goed op gepakt. Door goede communicatie tussen de teamgenoten is uiteindelijk een redelijk goed product opgeleverd. Verder hadden alle teamgenoten wel een eigen specialiteit, waardoor het werkverloop gemakkelijk ging en we veel van elkaar hebben kunnen leren.
Ook was het modelleren van het systeem erg goed verlopen, waardoor het maken van de uiteindelijke code erg gemakkelijk verliep.

Er zijn ook een aantal punten die wat minder goed ging, en waar nog ruimte is voor verbetering. Deze punten zijn onderdeel van het ontwikkelproces, maar kunnen ook een oplossing zijn op een probleem wat beter uitgewerkt had kunnen worden.

Als eerste moest helaas een van de teamgenoten om persoonlijke redenen het team verlaten. Hierdoor kwam het team enig sinds in tijdsnood, omdat dit teamlid al taken had gekregen bij het maken van de wasmachine. Hierdoor zijn niet aan alle eisen voldaan ten opzichte van de wasmachine. Het team had een beter product kunnen opleveren mits meer tijd te besteden was geweest.
Ten tweede had de communicatie met de teamleden beter gekund. Vaak werden de taken wel uitgevoerd, maar was niet duidelijk aan de andere teamleden vermeld dat de taak was uitgevoerd. Hierdoor was het elke dag de vraag hoever het project was.

Dit zijn dus de punten waarop verbetert kan worden.

7 Conclusies en aanbevelingen

In de conclusie en aanbevelingen wordt kort samengevat wat is besproken in dit verslag en wordt een conclusie getrokken uit de Requirements en Solution Architectuur. Ook worden aanbevelingen gegeven voor een eventuele opvolger.

Requirements architecture
Uit de Requirements Architecture kan worden geconcludeerd dat het systeem zo ontworpen is dat de gebruiker grootte vrijheid heeft is de opties voor het gebruik van de wasmachine. De gebruiker kan zelf een wasprogramma aanmaken en veranderen, een wasprogramma draaien vanaf een website en informatie verkrijgen over de wasmachine op de website. Ook bij het aanmaken van het wasprogramma heeft de gebruiker grootte vrijheid om zelf de temperatuur, toerental en waterniveau en vele andere opties in te stellen. Door deze belangrijke functionaliteiten kan de wasmachine op vele manieren worden toegepast.

Solution Architecture
Uit de Solution Architecture kan worden geconcludeerd dat het systeem efficiëntie is opgebouwd en gemakkelijk te bedienen is voor de gebruiker. De efficiëntie is vooral terug te zien in het gebruik van de verschillende taken, die op effectieve wijze met elkaar communiceren. Ook worden veel voorkomende problemen voorkomen door de synchronisatie methodes die zijn toegepast tussen de taken. Ook hierdoor kunnen de taken snel en effectief met elkaar communiceren.

Aanbevelingen
Om de wasmachine verder te realiseren en het een succesvol product te maken wordt aanbevolen om veel te testen met de wasmachine, de Raspberry Pi en de webserver. Omdat dit team niet genoeg tijd en kennis heeft gehad om het product werkend te krijgen is dit het belangrijkste deel.
Verder kan veel uit de MoSCoW worden gehaald, mocht de ontwikkelaar het product verder willen ontwikkelen. Hierin staan een aantal eisen die het product sterkt zullen verbeteren.

8 Bronvermeldingen

Ooijen, W. "*TI C++ software rules"* (2014) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015-2016-TI-C++-software-rules-1-0.pdf

Ooijen, W. "*Software Engineering in C++"*(2013) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/50/TCTI-V2CPSE1-15/Studiemateriaal/2015-2016-V2SECP1-reader.docx

Wensink, M. / W.v.Ooijen,"*Realtime System Programming*" (2013) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/8/TCTI-V2RTSP1-10/default.aspx?RootFolder=%2ffnt%2f8%2fTCTI-V2RTSP1-10%2fStudiemateriaal%2freader&FolderCTID=&View=%7b2C0A9E64-0ABB-4B83-94CA-36695FA8F7CE%7d

Wensink, M.*"Beschrijving wasmachine-emulator"* (2015) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/Beschrijving%20wasmachine-emulator.pdf

Wensink, M."*2015-2016-V2TH06 notes"*(2015) Verkregen van:
https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2THO6-14/Studiemateriaal/2015-2016-V2TH06-notes.pdf

9 Bijlagen

9.1 Interview verslag

Praat: Daniel ***Noteert: Wilco***

Inleiding:

Eerst voorstellen en rollen vermelden.
Vermelden van het doel van het interview.
(Vragen of er bezwaar is tegen het gebruik van opnameapparatuur.)
Agenda bespreken.
Vermelden wat gaat gebeuren met de verkregen informatie.

Wouter van Ooijen is bereid het technisch verslag te controleren.

Kern:

Praat: Robin & Daniel ***Noteert: Wilco & Wouter***

- **Webbrowser interface:**

Wat moet de gebruiker kunnen instellen bij het gebruik van de website?
Het huidige wasprogramma dat bezig is kan op elk moment verlengd of verkort worden, "met een draai aan de knop…"
De gebruiker moet zelf een wasprogramma naar keuze kunnen samen stellen en opslaan of uit een van de standaard wasprogramma's die al aanwezig zijn kiezen.
Deze zelf samengestelde wasprogramma's kunnen gewijzigd worden, uiteraard moeten deze wasprogramma's ook weer later na opslaan geladen en gebruikt kunnen worden.
Verschillende programma's voor verschillende situaties.
Back to factory settings reset, zodat alleen de standaard programma's aanwezig zijn.

Welke gegevens dienen met het interface te worden weergegeven?
De gebruiker moet een tijdlijn kunnen zien waarop zichtbaar is in welk stadium het wasprogramma zich bevindt, wat al klaar is en wat er nog gaat gebeuren. Ook de eindtijd / tijd hoe lang het wasprogramma nog bezig is moet zichtbaar zijn voor de gebruiker.
De temperatuur kan zichtbaar zijn maar dit is niet van belang.

Wat voor feedback krijgt de gebruiker tijdens het gebruiken van de website?
Het is handig voor de gebruiker om feedback te krijgen in bijvoorbeeld de vorm van een berichtje of geluid, iets zoals een sms bericht. Dit kan bijvoorbeeld gebeuren op het moment dat een wasprogramma klaar is.

Praat: Robin & Wilco ***Noteert: Daniel & Wouter***

- **Functionaliteiten Wasmachine.**

Voor wie is de wasmachine bedoeld? (studenten? families?)
De wasmachine is bedoeld voor de gewone huisvrouw maar heeft ook de mogelijkheid om gespecialiseerde wasprogramma's te gebruiken, bijvoorbeeld erg grote wasopdrachten of een specifiek soort was zoals een bepaald soort werk-kleding van een bedrijf.

Wat zijn de eisen zijn voor een wasprogramma?
Een wasprogramma moet zijn opgedeeld in fases. Een aantal van die fases zijn: water naar de juiste temperatuur brengen, wel of niet zeep toevoegen / een van de soorten zeep, wel of

niet (afval)water wegpompen, wel of niet draaien, spoelen, soorten zeep voor bijvoorbeeld glans of hoofdwas.

Moet een wasprogramma kunnen worden gestopt of gepauzeerd terwijl het bezig is?
Ja, met slidder om naar een volgend of vorige punt te gaan in het wasprogramma. Na pauzeren moet er verder worden gegaan op dat punt waar gepauzeerd was.

Wat voor was-programma's moeten aanwezig zijn in het systeem?
Voor de standaard wasprogramma's moet er worden gekeken bij de concurrentie, en daar inspiratie van uit opdoen.
Er moeten een aantal standaard wasprogramma's aanwezig zijn:
Witte was
Wol
Kwetsbare was.
Verder moet de gebruiker zelf een specifiek wasprogramma kunnen opbouwen.

Op welke aspecten van de wasmachine moeten er beveiligingen en waarschuwingen aanwezig zijn? zoals bijvoorbeeld een begrenzing op het verwarmen van het water.
Maximale temperatuur is niet echt nodig doordat water niet warmer kan worden dan 100 °C en niet zomaar kan gaan bevriezen.. Wel beveiliging om de deur te openen als er nog water in de machine zit en het verwarming-element mag niet aan als er geen water in zit.
Pompen laten stoppen op het moment dat er geen water meer in de machine zit.

Praat: Wouter ***Noteert: Daniel & Robin***
Server:
Wat zijn de functionaliteiten van de server?
Server is onderdeel van de wasmachine, dus die gaat aan en uit als de wasmachine aan of uit gaat.

Hoeveel wasmachine's moeten per server verwerkt kunnen worden?? (WAT? Bevat een server slechts één wasmachine of meerdere?)
Elke wasmachine heeft zijn eigen server.
Geen eis om meerdere machine's op een server te verbinden. Het product hoeft alleen te functioneren via een html site.

Praat: Daniel ***Noteert: Wilco***
Afsluiting:
Bedanken van de productleider.
Vragen naar belangstelling voor het eindrapport
Ja, er is belangstelling

Vragen naar contract mogelijkheden.
Contract is mogelijk via email.

9.2 Requirements

2.1 Must haves

- De gebruiker moet via de server verbinding kunnen maken met de machine.
- De gebruiker moet een wasprogramma kunnen kiezen.
- De gebruiker moet een wasprogramma kunnen aanzetten/pauzeren/stoppen.
- De machine moet een wasprogramma kunnen draaien.

- De fabrikant moet een wasprogramma kunnen samenstellen.
  - De fabrikant moet minstens 1 wasprogramma meeleveren.
  - Een wasprogramma moet uit minstens 1 fase bestaan.
- De machine moet beveiligd worden tegen de onkunde van de gebruiker.
  - De machine moet niet geopend kunnen worden wanneer een wasprogramma draait.
  - Beveiliging om het verwarming's element aan te zetten als er geen water in de machine aanwezig is.
  - Beveiliging om het pompen te laten stoppen als er geen water meer aanwezig is.

2.2 Should haves

- De gebruiker zou een wasprogramma moeten kunnen samenstellen en opslaan.
  - De gebruiker zou een eerder zelf gemaakt wasprogramma moeten kunnen wijzigen.
- De gebruiker zou feedback in de vorm van een bericht of geluid kunnen krijgen door de webserver, bijvoorbeeld op het moment dat een wasprogramma klaar is.
- De gebruiker zou een visuele weergave van het huidige wasprogramma en de status daarvan kunnen zien.
- De gebruiker zou de eindtijd en de tijdsduur moeten kunnen zien op het web interface.
- De gebruiker zou met een schuifbalk door het wasprogramma moeten kunnen lopen.
- Er zouden een aantal standaard wasprogramma's aanwezig moeten zijn: Witte was, wol en kwetsbare was.
- Een wasprogramma zou de volgende fases moeten bevatten: water naar de juiste temperatuur brengen, een van de soorten zeep toevoegen, (afval)water wegpompen, draaien en spoelen.

2.3 Could haves

- Om verbinding te maken met de server van de wasmachine kan de gebruiker een wachtwoord worden instellen.
- De gebruiker zou een tijdlijn kunnen zien waarop zichtbaar is in welk stadium het wasprogramma zich bevind, wat al klaar is en wat er nog gaat gebeuren.
- De gebruiker zou de optie kunnen hebben om het wasmachine systeem terug naar factory settings te resetten. (zelf gemaakte wasprogramma's verwijderen en standaard houden)
- De gebruiker zou niet essentiële informatie zoals bijv. de temperatuur kunnen zien.

2.4 Will-not haves

- Er zal geen detectie zijn voor teveel was in de trommel.
- Er zal geen domeinnaam voor de webinterface geregistreerd worden.
- Er zullen geen thema's voor de webinterface zijn. Dit houd in dat het eindproduct één vaste opmaak zal hebben, zonder optie deze te veranderen.
- Er zal geen functionaliteit worden geïmplementeerd om vanuit één webinterface meerdere wasmachines aan te sturen.

9.3 Requirements Architectuur

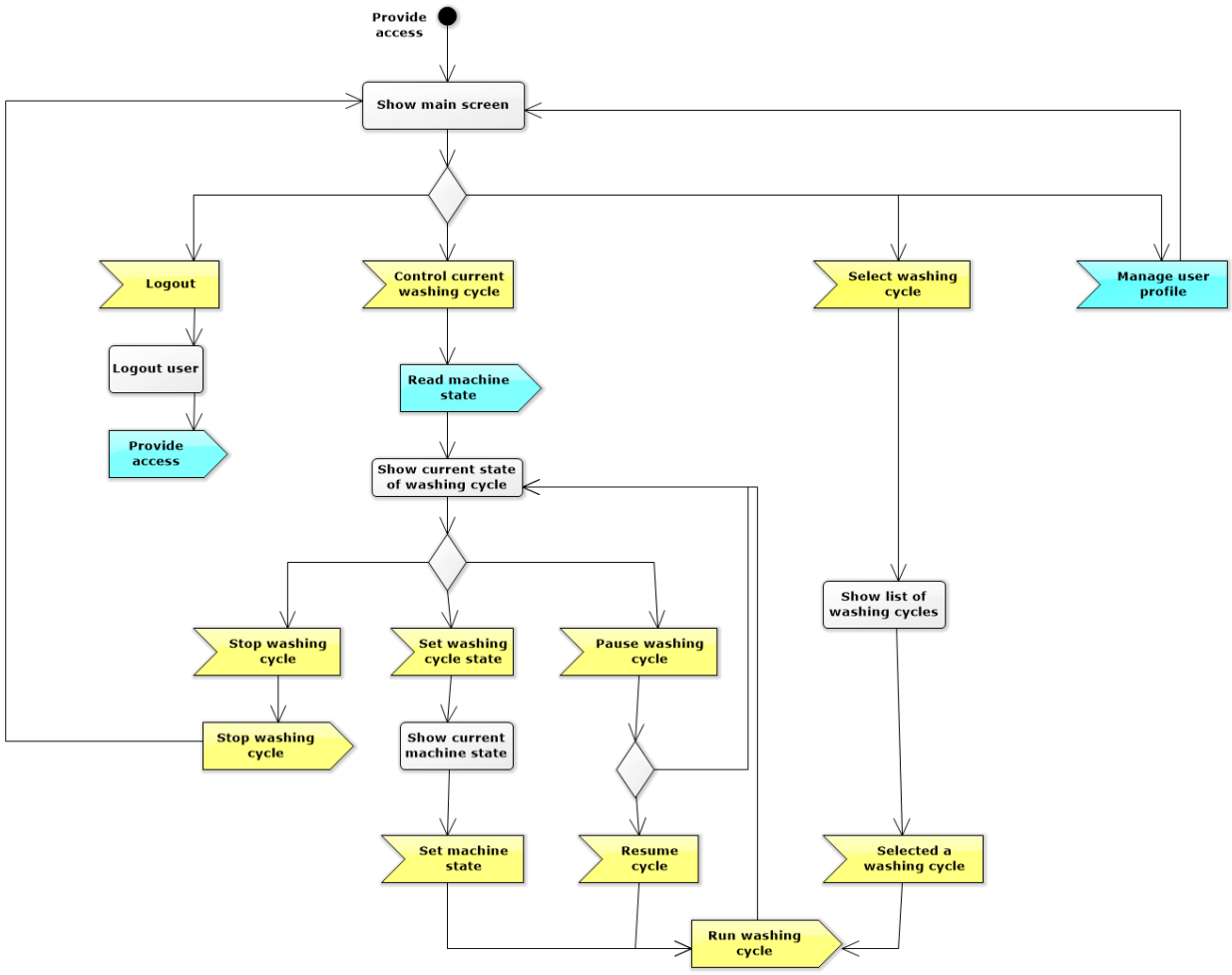
Activity Diagrams:

In dit hoofdstuk worden de activity Diagrams die horen bij de use-cases weergegeven met een korte beschrijving er bij. Een activity Diagram geeft zo duidelijk mogelijk weer wat de flow van (een deel van) het systeem is.

Control Washing Cycle:

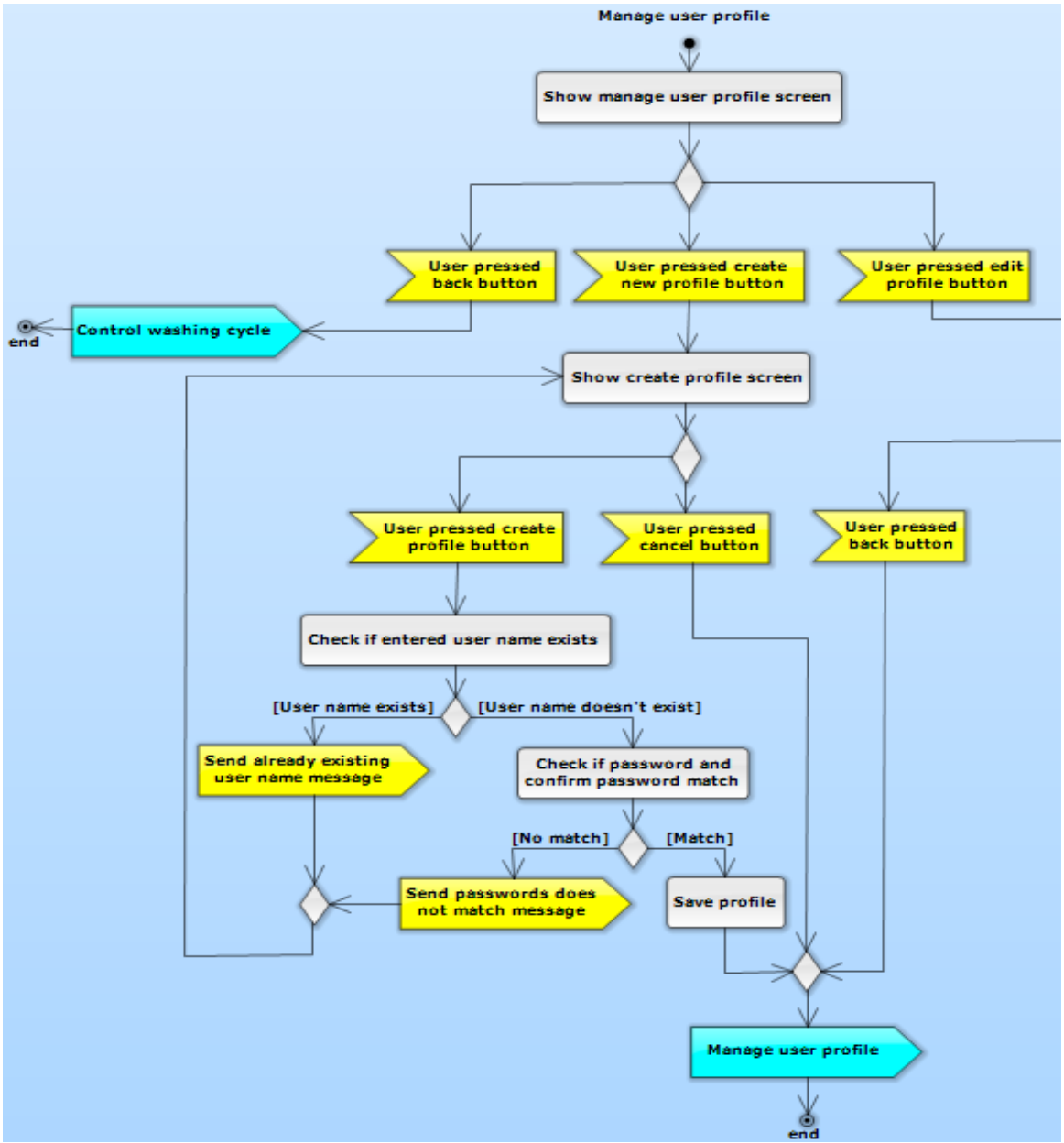
In de control washing cycle kan de gebruiker een wasprogramma kiezen en controleren. Onder controleren wordt verstaan dat de gebruiker het programma kan stoppen, pauzeren en weer verder kan laten gaan.

Als eerst wordt het hoofdscherm laten zien, waarin de gebruiker kan kiezen om zijn profiel te wijzigen, en dus ook zijn wasprogramma's. Verder kan de gebruiker kiezen om de het huidige programma te controleren of om een nieuw programma op te staten.



Manage User Profile, Deel 1:
In het Manage User Profile kan de gebruiker een nieuw profiel aanmaken of een bestaande beheren. Onder beheren valt het wijzigen van het wachtwoord van het huidige profiel of het toevoegen/wijzigen van een wasprogramma dat gekoppeld is aan het huidige profiel.

Eerst word het ‘Manage user profile scherm’ zichtbaar, waarin de gebruiker kan kiezen om een nieuw profiel te maken, zijn huidige profiel te wijzigen of terug te gaan naar het hoofdscherm.
Als er een nieuw profiel word aangemaakt moet de gebruiker in het ‘create profile scherm’ een gebruikersnaam en wachtwoord invoeren voor het nieuwe profiel. Hierna zal de gebruiker op de ‘create profile button’ moeten klikken om de ingevoerde gegevens te laten controleren en een nieuw profiel aan te maken. Of de gebruiker klikt op de ‘cancel button’ om terug te gaan naar het ‘Manage user profile scherm’.



Manage User Profile, Deel 2:

Als de gebruiker in het ‘Manage user profile scherm’ op de ‘edit profile button’ klikt word het ‘edit profile scherm’ geladen. In dit scherm kan de gebruiker op volgende 4 buttons klikken:

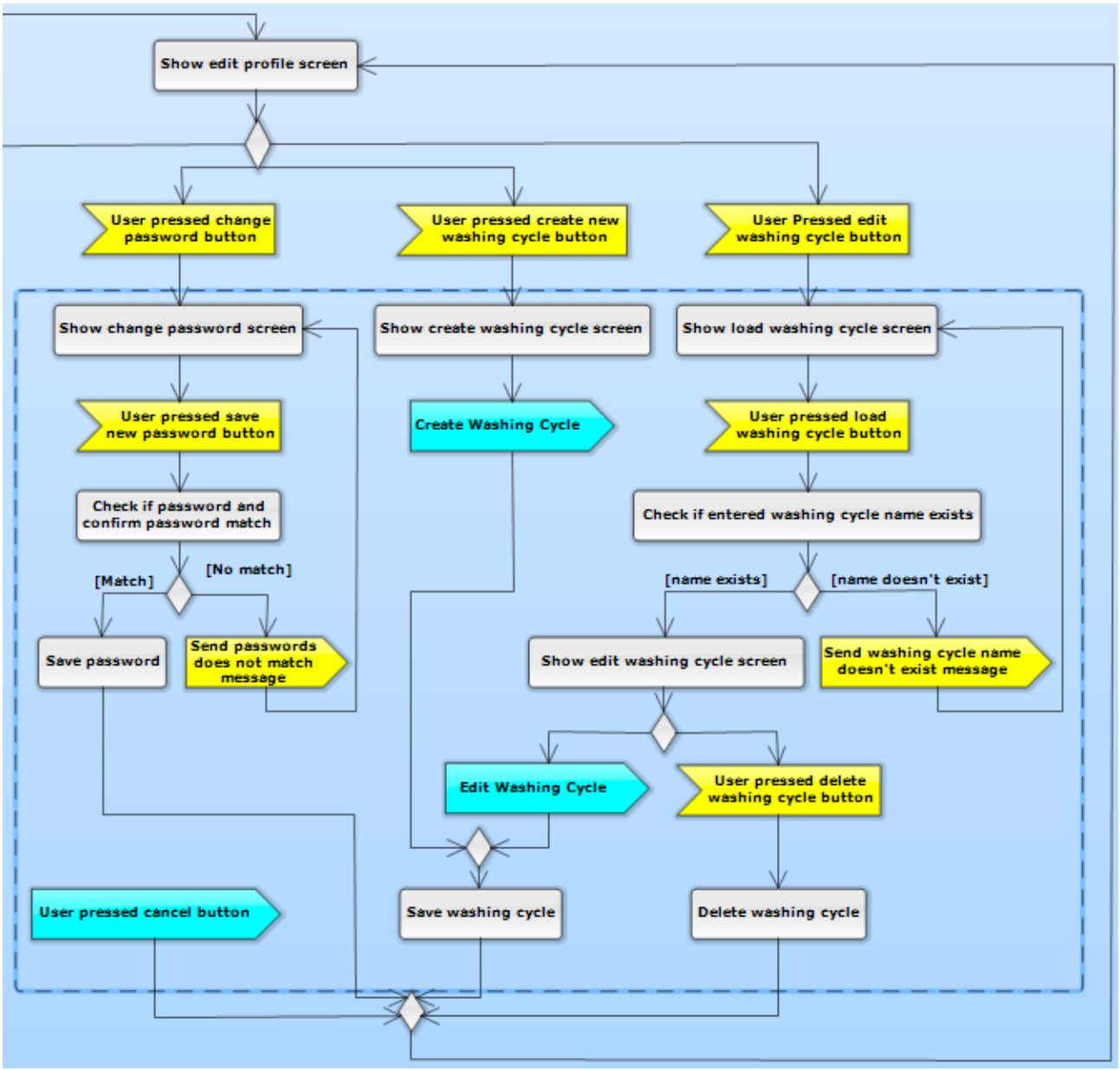
--De ‘back button’ zorgt dat het ‘Manage user profile scherm’ weer geladen word.

--De ‘change password button’ laad het ‘change password scherm’ waarin de gebruiker het wachtwoord van zijn/haar huidige actieve profiel kan wijzigen.

--De ‘create washing cycle button’ dat het ‘create washing cycle scherm’ laad waarin de gebruiker een nieuw wasprogramma kan samenstellen en opslaan in zijn/haar huidige profiel.

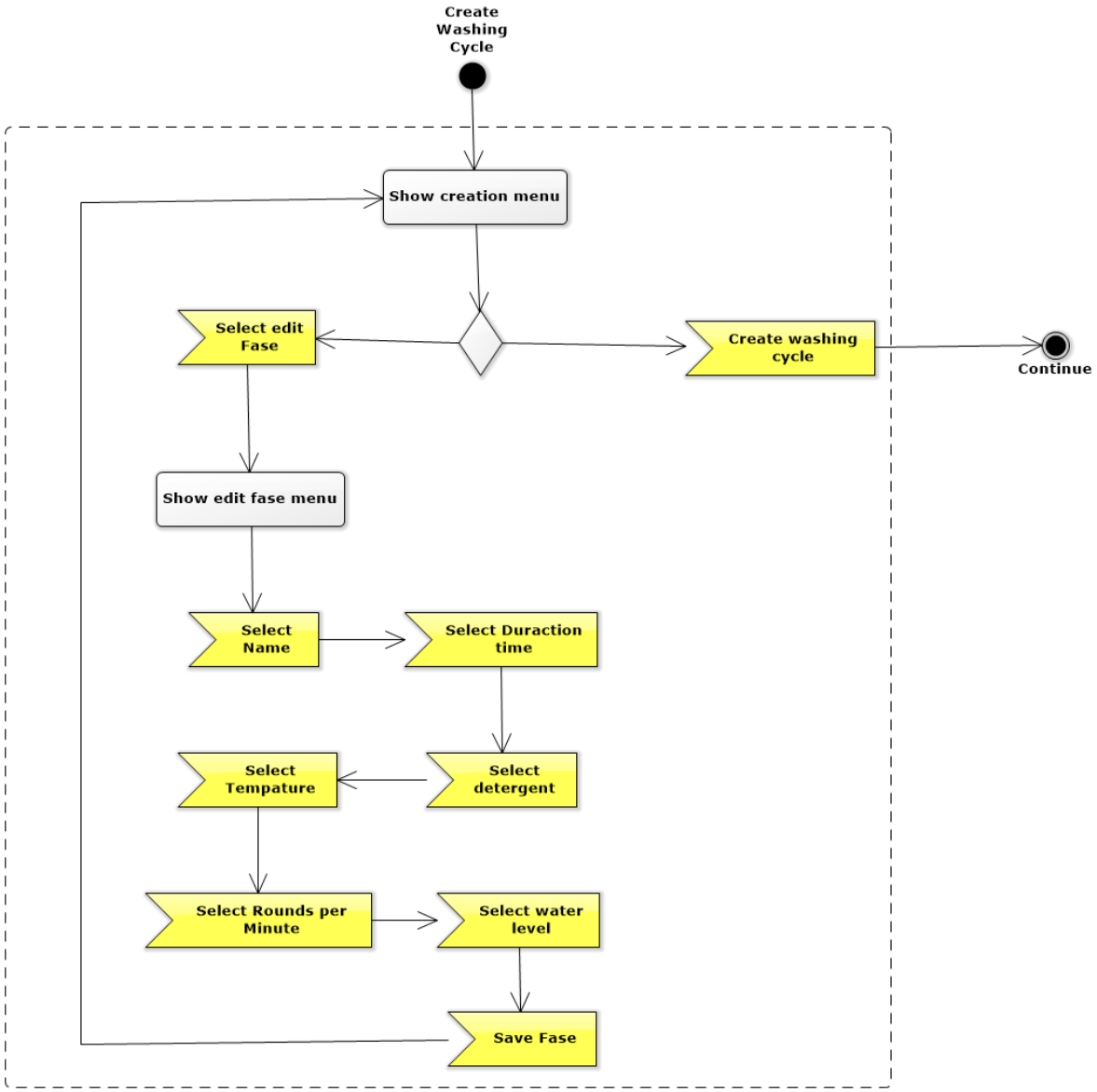
--En als laatste de ‘edit washing cycle button’ die het ‘load washing cycle scherm’ laat. In dit scherm kan de gebruiker een van zijn bestaande wasprogramma’s selecteren, als de gebruiker hierna op de ‘load washing cycle button’ klikt word het ‘edit washing cycle scherm’ geladen waar in de gebruiker het gekozen wasprogramma kan wijzigen of verwijderen.

Als er ergens in de schermen/menu’s die geladen worden na het ‘edit profile scherm’ op een cancel button word gedrukt zal het ‘edit profile scherm’ opnieuw geladen worden.



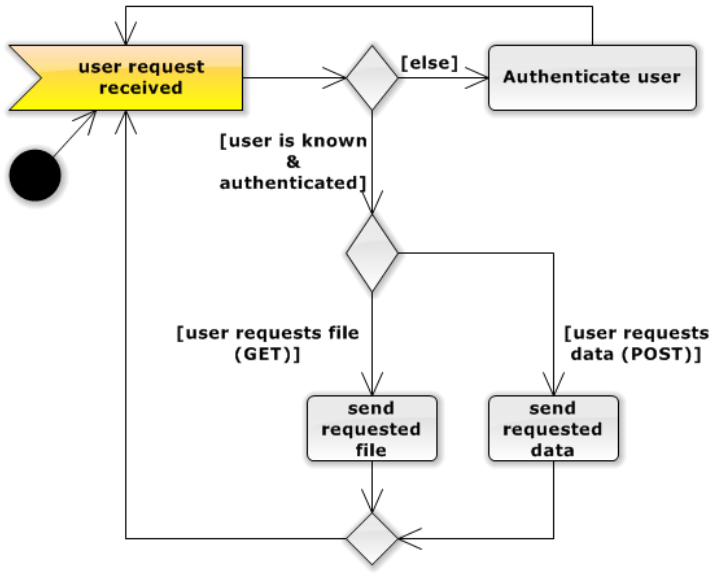
Create Washing Cycle

In create washing cycle wordt de gebruiker instaat gesteld om een wasprogramma aan te maken. De gebruiker kan zelf een aantal fases instellen. Deze fases worden dan na elkaar in het wasprogramma geplaatst.
In een fase kan de naam, de tijd, het wasmiddel, de temperatuur, het toerental en de hoeveelheid was worden ingesteld.



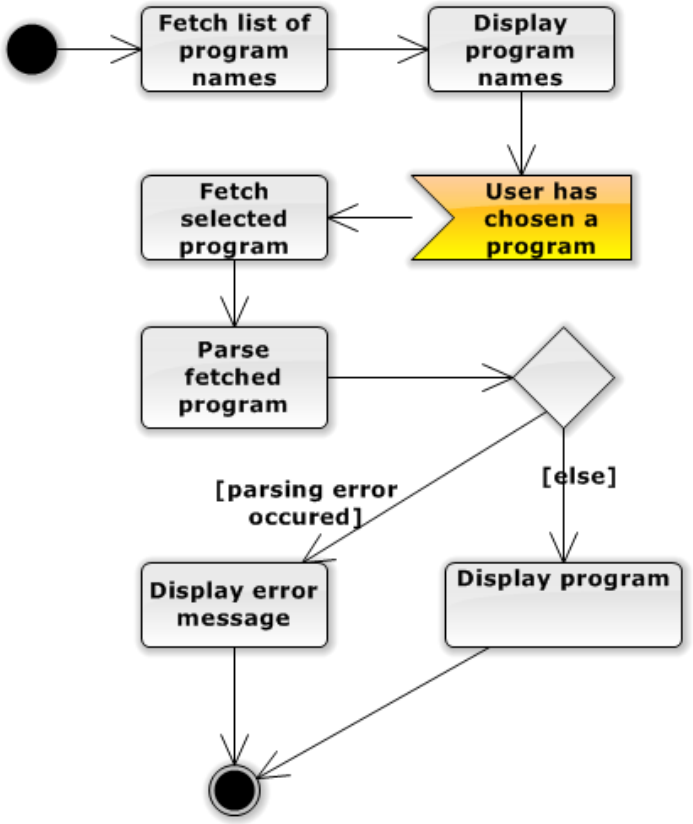
Display Machine State

Door het volbrengen van deze use case kan de wasmachine de afbeeldingen en webpagina's doorsturen die nodig zijn om de gebruiker te laten communiceren met de wasmachine, en ook de huidige statistieken zoals watertemperatuur en toerental doorgeven naar de browser van de gebruiker.



Load Saved State

Deze use case moet worden voltooid wanneer de gebruiker ofwel een programma wil selecteren voor gebruik, of voor het maken van aanpassingen. Na het uitvoeren van deze use case is er een programma ingeladen en wordt dit weergegeven aan de gebruiker.

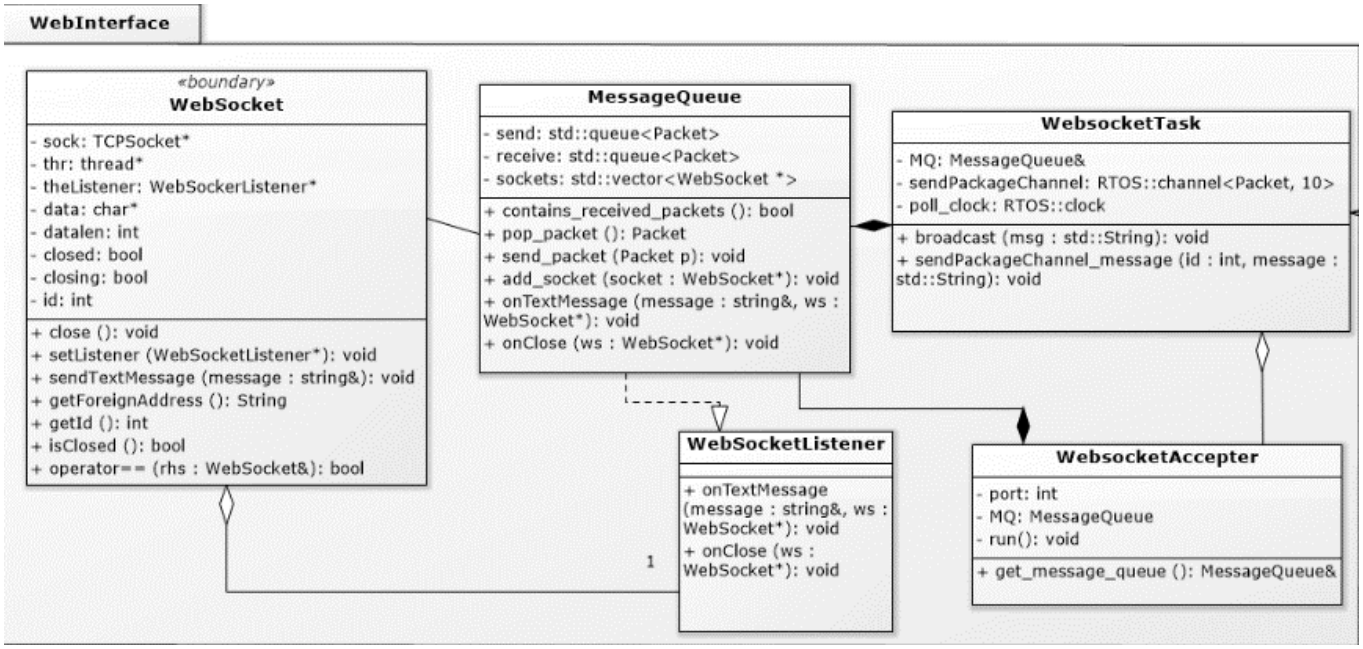


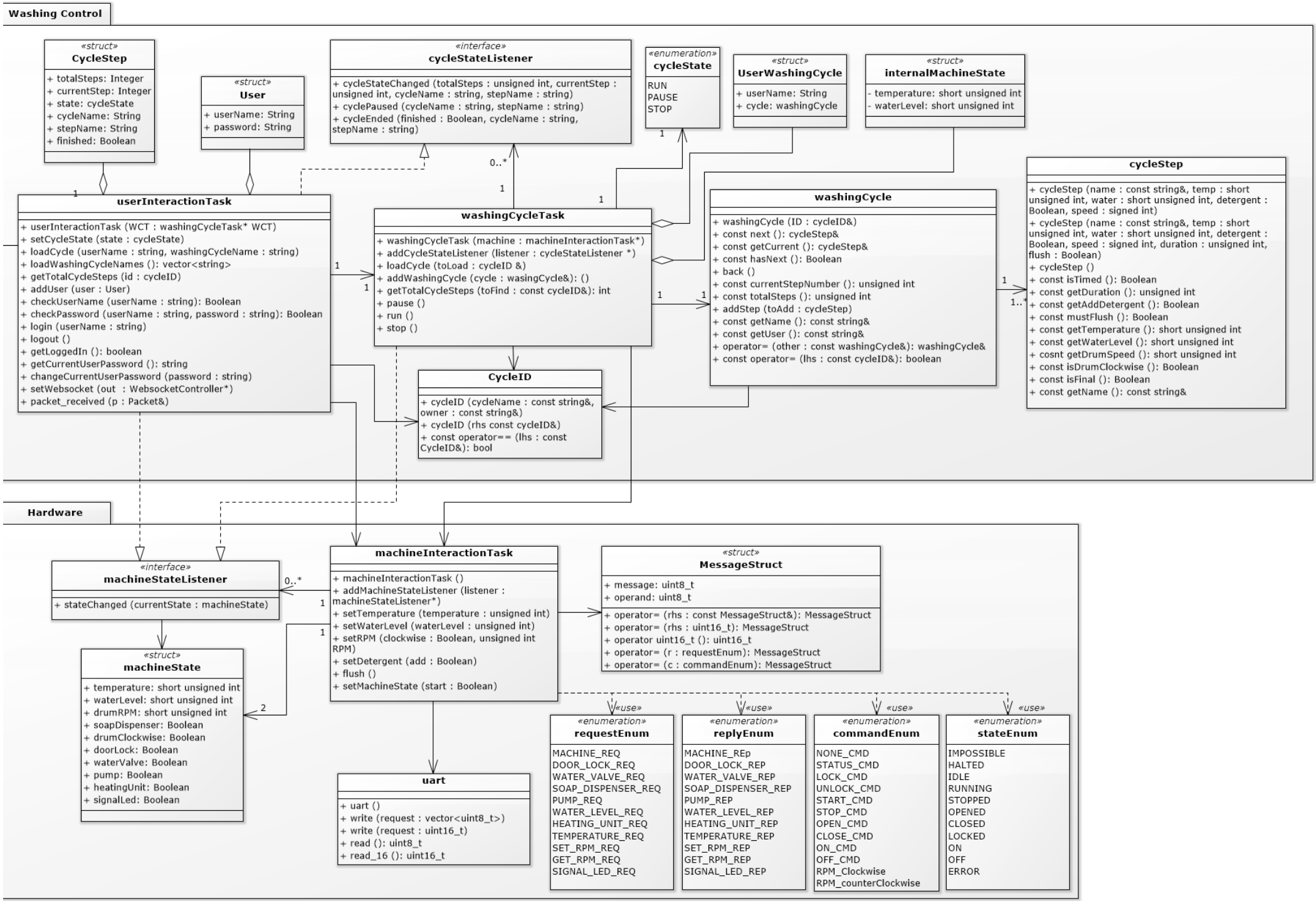
Constraints

In de constraints worden alle niet functionele eisen aan het systeem behandeld. Het systeem moet aan deze eisen voldoen.

Type	*Identificatie*	*Beschrijving*	*Criterium*	Verificatie
Performance	Reactietijd	Tijd waarbinnen moet worden gereageerd op signalen vanaf de webbrowser	binnen 200 ms	Meet de tijdsduur door middel van een timer binnen de code.
	Update van waarden	De tijd waarbinnen de informatie op het scherm wordt herladen.	binnen 1 seconde	
Accuracy	Login informatie	De foutmarge waarin de gebruikersnaam en wachtwoord geaccepteerd worden	0 afwijkingen	Een log file bijhouden.
	Temperatuur	De foutmarge waarin de temperatuur van de wasmachine mag afwijken van de voorgeprogrammeerde temperatuur	maximaal 2 graden Celsius	Systeem laten uitprinten en zelfstandig waarnemen
	Toerental	De foutmarge waarin het toerental van de wasmachine mag afwijken van het voorgeprogrammeerde toerental	maximaal 50 toeren	
	Waterlevel	De foutmarge waarin het waterlevel van de wasmachine mag afwijken van het voorgeprogrammeerde waterlevel	maximaal 5 %	
Usability	Webinterface	Door welke doelgroep het webinterface gebruikt moet kunnen worden.	De huisvrouw van onze klant moet het kunnen gebruiken	Testen door het te laten gebruiken door een aantal verschillende personen.
Learnability	Webinterface	Hoeveel tijd het kost om het webinterface te leren gebruiken	Binnen 30 minuten	

9.4 Volledig Klassen Diagram




Washing Control
«struct»
CycleStep
+ totalSteps: Integer
+ currentStep: Integer
+ state: cycleState
+ cycleName: String
+ stepName: String
+ finished: Boolean
«struct»
User
+ userName: String
+ password: String
«interface»
cycleStateListener
+ cycleStateChanged (totalSteps : unsigned int, currentStep : unsigned int, cycleName : string, stepName : string)
+ cyclePaused (cycleName : string, stepName : string)
+ cycleEnded (finished : Boolean, cycleName : string, stepName : string)
«enumeration»
cycleState
RUN
PAUSE
STOP
«struct»
UserWashingCycle
+ userName: String
+ cycle: washingCycle
«struct»
internalMachineState
- temperature: short unsigned int
- waterLevel: short unsigned int
userInteractionTask
+ userInteractionTask (WCT : washingCycleTask* WCT)
+ setCycleState (state : cycleState)
+ loadCycle (userName : string, washingCycleName : string)
+ loadWashingCycleNames (): vector<string>
+ getTotalCycleSteps (id : cycleID)
+ addUser (user : User)
+ checkUserName (userName : string): Boolean
+ checkPassword (userName : string, password : string): Boolean
+ login (userName : string)
+ logout ()
+ getLoggedIn (): boolean
+ getCurrentUserPassword (): string
+ changeCurrentUserPassword (password : string)
+ setWebsocket (out : WebsocketController*)
+ packet_received (p : Packet&)
washingCycleTask
+ washingCycleTask (machine : machineInteractionTask*)
+ addCycleStateListener (listener : cycleStateListener *)
+ loadCycle (toLoad : cycleID &)
+ addWashingCycle (cycle : wasingCycle&): ()
+ getTotalCycleSteps (toFind : const cycleID&): int
+ pause ()
+ run ()
+ stop ()
washingCycle
+ washingCycle (ID : cycleID&)
+ const next (): cycleStep&
+ const getCurrent (): cycleStep&
+ const hasNext (): Boolean
+ back ()
+ const currentStepNumber (): unsigned int
+ const totalSteps (): unsigned int
+ addStep (toAdd : cycleStep)
+ const getName (): const string&
+ const getUser (): const string&
+ operator= (other : const washingCycle&): washingCycle&
+ const operator= (lhs : const cycleID&): boolean
cycleStep
+ cycleStep (name : const string&, temp : short unsigned int, water : short unsigned int, detergent : Boolean, speed : signed int)
+ cycleStep (name : const string&, temp : short unsigned int, water : short unsigned int, detergent : Boolean, speed : signed int, duration : unsigned int, flush : Boolean)
+ cycleStep ()
+ const isTimed (): Boolean
+ const getDuration (): unsigned int
+ const getAddDetergent (): Boolean
+ const mustFlush (): Boolean
+ const getTemperature (): short unsigned int
+ const getWaterLevel (): short unsigned int
+ cosnt getDrumSpeed (): short unsigned int
+ const isDrumClockwise (): Boolean
+ const isFinal (): Boolean
+ const getName (): const string&
CycleID
+ cycleID (cycleName : const string&, owner : const string&)
+ cycleID (rhs const cycleID&)
+ const operator== (lhs : const CycleID&): bool
Hardware
«interface»
machineStateListener
+ stateChanged (currentState : machineState)
machineInteractionTask
+ machineInteractionTask ()
+ addMachineStateListener (listener : machineStateListener*)
+ setTemperature (temperature : unsigned int)
+ setWaterLevel (waterLevel : unsigned int)
+ setRPM (clockwise : Boolean, unsigned int RPM)
+ setDetergent (add : Boolean)
+ flush ()
+ setMachineState (start : Boolean)
«struct»
MessageStruct
+ message: uint8_t
+ operand: uint8_t
+ operator= (rhs : const MessageStruct&): MessageStruct
+ operator= (rhs : uint16_t): MessageStruct
+ operator uint16_t (): uint16_t
+ operator= (r : requestEnum): MessageStruct
+ operator= (c : commandEnum): MessageStruct
«struct»
machineState
+ temperature: short unsigned int
+ waterLevel: short unsigned int
+ drumRPM: short unsigned int
+ soapDispenser: Boolean
+ drumClockwise: Boolean
+ doorLock: Boolean
+ waterValve: Boolean
+ pump: Boolean
+ heatingUnit: Boolean
+ signalLed: Boolean
uart
+ uart ()
+ write (request : vector<uint8_t>)
+ write (request : uint16_t)
+ read (): uint8_t
+ read_16 (): uint16_t
«use»
«enumeration»
requestEnum
MACHINE_REQ
DOOR_LOCK_REQ
WATER_VALVE_REQ
SOAP_DISPENSER_REQ
PUMP_REQ
WATER_LEVEL_REQ
HEATING_UNIT_REQ
TEMPERATURE_REQ
SET_RPM_REQ
GET_RPM_REQ
SIGNAL_LED_REQ
«enumeration»
replyEnum
MACHINE_REp
DOOR_LOCK_REP
WATER_VALVE_REP
SOAP_DISPENSER_REP
PUMP_REP
WATER_LEVEL_REP
HEATING_UNIT_REP
TEMPERATURE_REP
SET_RPM_REP
GET_RPM_REP
SIGNAL_LED_REP
«enumeration»
commandEnum
NONE_CMD
STATUS_CMD
LOCK_CMD
UNLOCK_CMD
START_CMD
STOP_CMD
OPEN_CMD
CLOSE_CMD
ON_CMD
OFF_CMD
RPM_Clockwise
RPM_counterClockwise
«enumeration»
stateEnum
IMPOSSIBLE
HALTED
IDLE
RUNNING
STOPPED
OPENED
CLOSED
LOCKED
ON
OFF
ERROR